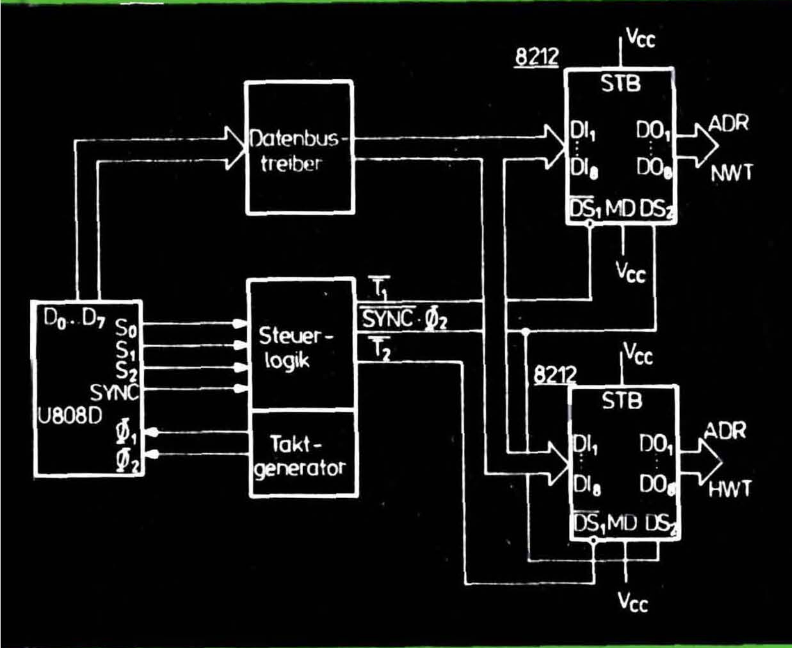
amateurreihe

# electronica




Barthold/Bäurich

203

## Mikroprozessoren — Mikroelektronische Schaltkreise und ihre Anwendung (Teil 2)

electronica · Band 203

HANS BARTHOLD
DR. HEINZ BÄURICH

# Mikroprozessoren – Mikroelektronische Schaltkreise und ihre Anwendung

**Teil 2:**
**Die Mikroprozessoren** *U 808 D*, *U 880 D* **und** *8080*



MILITÄRVERLAG
DER DEUTSCHEN DEMOKRATISCHEN
REPUBLIK

2. Auflage, 1982

Lizenz-Nr. 5 · LSV 3539
Lektor: Wolfgang Stammler
Zeichnungen: Johanna Goernemann
Typografie: Helmut Herrmann
Printed in the German Democratic Republic
Gesamtherstellung: Druckerei Märkische Volksstimme Potsdam
Redaktionsschluß: 15. Februar 1982
Bestellnummer: 746 417 9
DDR 1,90 M

# Inhaltsverzeichnis


| | | |
|---|---|---|
| **1.** | **Formelzeichen und Abkürzungen** | 5 |
| **2.** | **Der Mikroprozessorbaustein** *U 808 D* | 6 |
| 2.1. | Registerstruktur des Mikroprozessorbausteins *U 808 D* | 6 |
| 2.2. | Befehlsaufbau des Bausteins *U 808 D* | 9 |
| 2.2.1. | Befehlsstruktur | 9 |
| 2.2.2. | Adreßbildung (Bestimmung des zum Befehl gehörenden Operanden) | 9 |
| 2.3. | Zeitverhalten des Bausteins *U 808 D* | 10 |
| 2.4. | Befehlsabarbeitung | 13 |
| 2.5. | Befehlsliste *U 808 D* | 15 |
| 2.5.1. | Erläuterung der in der Befehlsliste verwendeten Abkürzungen | 15 |
| 2.5.2. | Beschreibung der Befehle des Mikroprozessors *U 808 D* | 16 |
| 2.5.2.1. | Transportoperationen | 16 |
| 2.5.2.2. | Rechen- und logische Operationen mit einem Operand | 17 |
| 2.5.2.3. | Rechen- und logische Operationen mit zwei Operanden | 18 |
| 2.5.2.4. | Sprungbefehle | 19 |
| 2.5.2.5. | Unterprogrammbefehle | 20 |
| 2.5.2.6. | Eingabe- und Ausgabebefehle | 21 |
| 2.5.2.7. | Steuerbefehle | 22 |
| 2.6. | INTERRUPT | 23 |
| 2.7. | Starten des Prozessors *U 808 D* | 24 |
| 2.8. | Beschreibung der Anschlüsse des Bausteins *U 808 D* | 26 |
| 2.9. | Anschluß externer Schaltkreise an den Prozessor *U 808 D* | 28 |
| **3.** | **Der Mikroprozessorbaustein** *U 880 D* | 32 |
| 3.1. | Registerstruktur des Mikroprozessorbausteins *U 880* | 32 |
| 3.2. | Befehlsaufbau des Bausteins *U 880* | 36 |

3.2.1. Befehlsstruktur ... 36
3.2.2. Adreßbildung ... 37
3.3. Zeitverhalten ... 38
3.4. Befehlsabarbeitung ... 44
3.5. Befehlsliste des Prozessors *U 880* ... 44
3.5.1. Verwendete Abkürzungen bei der Befehlsbeschreibung ... 44
3.5.2. Beschreibung der Befehle des Prozessors *U 880* 45
3.5.2.1. Transportbefehle ... 45
3.5.2.2. Rechen- und logische Operationen mit einem Operand ... 53
3.5.2.3. Rechen- und logische Operationen mit zwei Operanden ... 60
3.5.2.4. Rechen- und logische Operationen mit mehreren Operanden ... 63
3.5.2.5. Sprungbefehle ... 65
3.5.2.6. Unterprogrammbefehle ... 67
3.5.2.7. Ein- und Ausgabebefehle ... 69
3.5.2.8. Steuerbefehle ... 74
3.6. INTERRUPT ... 75
3.7. Starten des Prozessors *U 880* ... 78
3.8. Anschlüsse an den Baustein *U 880* ... 79

4. **Der Mikroprozessorbaustein** *8080* ... 82
4.1. Registerstruktur ... 83
4.2. Befehlsaufbau ... 83
4.3. Zeitverhalten ... 83
4.4. Anschlußsignale ... 85
4.5. Anschluß von Schaltkreisen an den Prozessor *8080* 88
4.6. Befehlsschlüssel des Prozessors *8080* ... 88
4.7. Programmunterbrechung ... 89

**Literaturverzeichnis** ... 90

**Anhang**
Befehlstabellen der Prozessoren *U 880, 8080, U 808 D* ... 91
Codierungstabelle der Befehle des Prozessors *U 808D* ... 102
Codierungstabelle der Befehle des Prozessors *8080* ... 104
Codierungstabelle der Befehle des Prozessors *U 880* ... 106
Gegenüberstellung des 1. Operationscodebytes der Mikroprozessoren *U 880, 8080, U 808 D* ... 114

# 1. Formelzeichen und Abkürzungen

Nachstehend sind die verwendeten Formelzeichen und Abkürzungen aufgeführt. Spezielle Zeichen, die nur wenig benutzt werden, sind im Text erläutert.

| | |
|---|---|
| ADR | Adresse |
| ADR-HWT oder $ADR_H$ | höherwertiger Adreßteil |
| ADR-NWT oder $ADR_L$ | niederwertiger Adreßteil |
| Index H | höherwertiger Teil eines 16-Bit-Wortes |
| Index L | niederwertiger Teil eines 16-Bit-Wortes |
| DB | Datenbus |
| $D_7 \ldots D_0$ | Bitstellen des Datenbus<br>$D_0$ = niederwertiges Bit |
| $A_{15} \ldots A_0$ | Bitstellen des Adreßbus<br>$A_0$ = niederwertiges Bit |
| SP | Stackpointer (Stackzeiger) |
| PC | Programm-Counter (Befehlszähler) |
| BR | Befehlsregister |
| $n$, d | eine 8-Bit-Zahl |
| $nn$ | eine 16-Bit-Zahl |
| $\Phi_1$, $\Phi_2$, $\Phi$ | Taktsignale |

# 2. Der Mikroprozessorbaustein *U 808 D*

Wie schon in Teil 1 erwähnt, beinhaltet der Mikroprozessor das Rechen- und Steuerwerk eines Rechners. Zum Einbau in Rechnersysteme verfügt er über Ein- und Ausgabesignale, mit deren Hilfe weitere Bausteine eines Rechners angeschlossen werden können. Die Ein- und Ausgangssignale des Mikroprozessors kann man unterteilen in

- Adreßsignale (Adreßbus),
- Datensignale (Datenbus),
- Steuer- und Meldesignale (Controlbus),
- Versorgungsspannungen (Taktsignale, Betriebsspannungen und Masseleitung).

Die Arbeitsweise des Mikroprozessors wird durch seinen Befehlsvorrat bestimmt. Den Befehlen entsprechen Signale, durch die die Inhalte der einzelnen internen Register untereinander transportiert werden. Während des Transports führen logische Schaltungen die einzelnen Operationen aus. Voraussetzung zum Verständnis des Befehlsschlüssels ist deshalb die Registerstruktur des Prozessors.

## 2.1. Registerstruktur des Mikroprozessorbausteins *U 808 D*

Bild 2.1 zeigt die Registerstruktur des Bausteins *U 808 D*. Der Mikroprozessor enthält einen internen 8-Bit-Bus, von dem aus alle Register zu erreichen sind. Von diesem Bus aus gehen die Daten über den externen Datenbus ($D_0 \ldots D_7$), der in beiden Richtungen betrieben werden kann, zu den angeschlossenen Schaltkreisen. Der externe Datenbus ist über Bustreiber mit dem internen 8-Bit-Bus verbunden.

An den internen 8-Bit-Bus sind angeschlossen:

- ein Notizspeicher, der aus 7 Registern (A, B, C, D, E, H, L) besteht (A = Akkumulatorregister);
- ein Kellerspeicher (STACK), der aus 7 Registern mit einer Länge von je 14 Bit besteht;

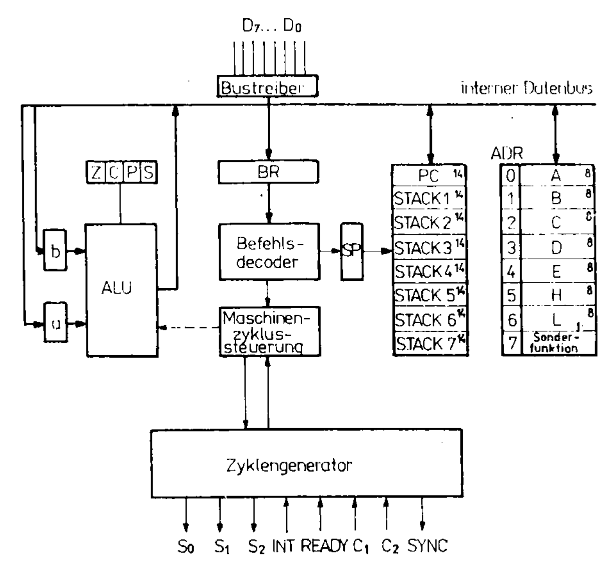



**Bild 2.1** Registerstruktur des Bausteins U *808 D* (statt $C_1$, $C_2$ am Zyklengenerator lies $\Phi_1$, $\Phi_2$)

- ein Befehlszähler (PC) der Länge 14 Bit;
- ein Befehlsregister zur Speicherung des Operationscodes eines Befehls;
- zwei dynamische Register a und b, die den Eingang in die arithmetisch-logische Einheit (ALU = *a*rithmetic-*l*ogic-*u*nit) darstellen.

  Die arithmetisch-logische Einheit ist wiederum mit dem Datenbus verbunden.

An das Befehlsregister ist ein Befehlsdecoder angeschlossen, dessen Ausgänge in Verbindung mit den Zeitsignalen in der Steuereinheit die Torungssignale für die Steuerung der Register und der arithmetisch-logischen Einheit bilden. Die Zeitsignale werden im Zustandsgenerator mit Hilfe der Taktfolgen $\Phi_1$ und $\Phi_2$ gebildet.

Ein Befehl wird innerhalb des Bausteins in folgender Reihenfolge verarbeitet:

1. Ausgabe der Adresse über den externen Datenbus (erst den niederwertigen Teil, danach den höherwertigen Teil);

2. Erhöhung des Befehlszählers um 1;
3. Einlesen des Operationscodes des Befehls in das Befehlsregister;
4. Entschlüsselung des Befehls und Bildung der Torungssignale in der Steuerschaltung;
5. Ausführung der Operation mit Hilfe der gebildeten Torungssignale (es werden die Inhalte der angesteuerten Register gelesen und über die eingestellten Datenwege transportiert). Benötigt man zur Ausführung des Befehls eine Zahl aus dem angeschlossenen Speicher, so wird zum Holen dieser Zahl die dazugehörige Adresse über den Datenbus ausgegeben und anschließend das entsprechende Byte über den Datenbus aus dem Speicher übernommen.

Zur Ausführung der Rechenoperationen und logischen Operationen enthält der Baustein einen Rechenwerksteil. Er besteht aus
- der arithmetisch-logischen Einheit (ALU);
- zwei Eingangsregistern a und b;
- den Flags bzw. Flip-Flop Z, C, P, S.

Die *Flags* haben im einzelnen folgende Aufgabe:

**Z (Zero-Flag)**
Z wird gleich 1 gesetzt, wenn das Ergebnis einer Rechenoperation 0 ist.

**C (Carry-Flag)**
C wird gleich 1 gesetzt, wenn bei der Addition ein Übertrag in die 8. Stelle auftritt oder wenn bei der Subtraktion ein „Borgen" von der 8. Stelle notwendig ist (die 8. Stelle entspricht dem Stellenwert $2^8$).

**P (Parity-Flag)**
P wird gleich 1 gesetzt, wenn das Ergebnis einer Operation eine geradzahlige Anzahl der Ziffer 1 enthält.

**S (Sign-Flag)**
S wird gleich 1 gesetzt, wenn das Ergebnis einer Operation negativ ist.

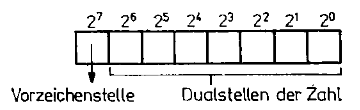



**Bild 2.2** Aufbau eines Zahlworts im Baustein *U 808 D*

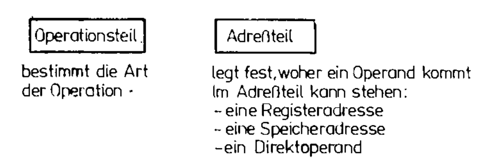



Bild 2.3 Grundsätzlicher Aufbau eines Befehls

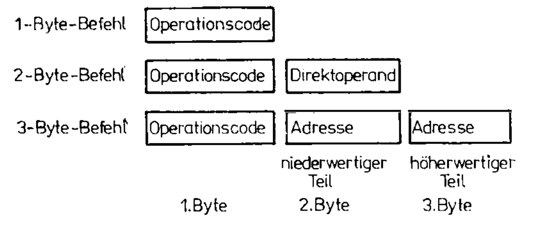



Bild 2.4 Darstellung eines Befehls im *U 808 D*

Der Baustein arbeitet mit einer Zahlendarstellung *Festkomma-Zweierkomplement mit 8 Dualstellen*. Der Stellenwert $2^7$ entspricht dem Vorzeichen (Bild 2.2).

## 2.2. Befehlsaufbau des Bausteins *U 808 D*

### 2.2.1. Befehlsstruktur

Ein Befehl besteht aus *Operationsteil* und *Adreßteil* (Bild 2.3). Zur Darstellung eines Befehls im *U 808 D* werden 1 bis 3 Byte benötigt. Davon erfordert der Operationscode 1 Byte und der Adreßteil 1 bis 2 Byte (Bild 2.4).

### 2.2.2. Adreßbildung (Bestimmung des zum Befehl gehörenden Operanden)

Für die Bestimmung eines Operanden gibt es folgende Möglichkeiten
– Direktoperand

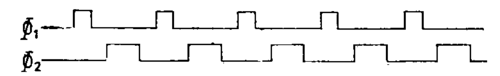



Bild 2.5 Taktsignale $\Phi_1$ und $\Phi_2$ zur Steuerung des Mikroprozessors *U 808 D*

Der zum Befehl gehörende Operand steht im Anschluß an den Operationscode. (1. Byte: Befehl; 2. Byte: Operand).

- Indirekte Adressierung

  Die Adresse ADR des Operanden steht in einem speziellen Register.

  ADR = ⟨Register⟩

  Als Register dient das Registerpaar HL.

- Registeroperand

  Der Operand steht in einem im Befehl angegebenen Register.

## 2.3. Zeitverhalten des Bausteins *U 808 D*

Der Mikroprozessor *U 808 D* wird durch zwei gegeneinander versetzte Impulsfolgen (Bild 2.5) gesteuert.

Die Abarbeitung eines Befehls unterteilt sich im Baustein in mehrere Maschinenzyklen. Ein Maschinenzyklus ist wiederum in einzelne Zeitzustände aufgeteilt. Ein Zeitzustand hat die Länge von 2 aufeinanderfolgenden Perioden des Taktes $\Phi_1$. Zur Unterscheidung der beiden Perioden eines Zustands sendet der Mikroprozessor während der 1. Periode eines Zustands das Signal SYNC (Bild 2.6).

Die möglichen Zustände eines Maschinenzyklus und ihre grundsätzliche Bedeutung sind:

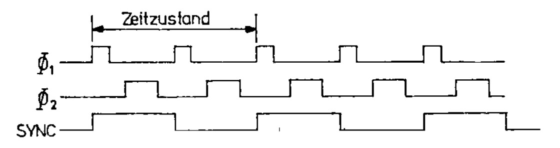



**Bild 2.6 Aufteilung der Taktperiode in Zeitzustände beim *U 808 D*. Ein Zeitzustand hat die Länge von 2 aufeinanderfolgenden Perioden des Taktes $\Phi_1$. Während der 1. Periode sendet der Prozessor das Signal SYNC aus**

$T_1$ Während $T_1$ wird der niederwertige Teil der Adresse über den Bus ausgegeben.

$T_{1I}$ Der Zustand $T_{1I}$ tritt nur bei INTERRUPT auf. Er steht dann an Stelle von $T_1$. Auf seine Bedeutung wird in Abschnitt 2.6. eingegangen.

$T_2$ Während $T_2$ wird der höherwertige Teil der Adresse ausgegeben.

WAIT Im Zustand WAIT wartet der Mikroprozessor auf die Bereitschaft externer Einheiten.

$T_3$ Während $T_3$ geschieht die Ein- und Ausgabe von Daten.

STOP Der Stoppzustand tritt nach jedem HALT-Befehl auf.

$T_4$, $T_5$ Während $T_4$ und $T_5$ werden die Befehle abgearbeitet.

Über die Signale $S_0$, $S_1$ und $S_2$ wird der externen Schaltung der Zustand angezeigt, in dem sich der Mikroprozessor gerade befindet. Aus Tabelle 2.1. ist die Belegung der Signale $S_0$, $S_1$, $S_2$ für die einzelnen Zustände zu ersehen.

Bild 2.7 zeigt das Taktdiagramm der Zeitstufen eines Maschinenzyklus und den dazugehörigen Pegel der Anschlüsse SYNC, $S_0$, $S_1$, $S_2$.

Aus Bild 2.8 geht hervor, unter welchen Bedingungen sich der Übergang von einem in den nächsten Zustand vollzieht und in welcher Reihenfolge die Zustände auftreten können.

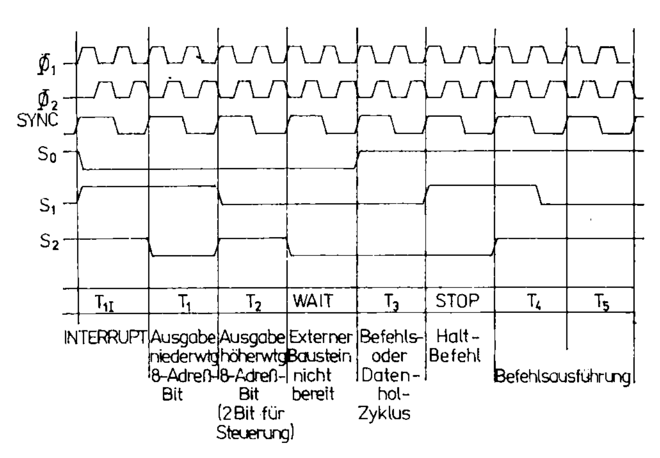



**Bild 2.7** **Taktdiagramm der Zeitstufen eines Maschinenzyklus des** ***U 808 D***

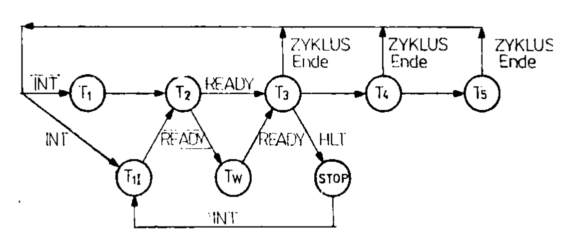



Bild 2.8 Blockdiagramm für die Zustandsübergänge des Bausteins *U 808 D*

**Tabelle 2.1.** Zustandstabelle der Statussignale $S_0$, $S_1$, $S_2$ des Prozessors *U 808 D*

| Zustand | $S_0$ | $S_1$ | $S_2$ | Code |
|---|---|---|---|---|
| $T_1$ | 0 | 1 | 0 | 2 |
| $T_{1I}$ | 0 | 1 | 1 | 3 |
| $T_2$ | 0 | 0 | 1 | 1 |
| WAIT | 0 | 0 | 0 | 0 |
| $T_3$ | 1 | 0 | 0 | 4 |
| STOP | 1 | 1 | 0 | 6 |
| $T_4$ | 1 | 1 | 1 | 7 |
| $T_5$ | 1 | 0 | 1 | 5 |

Liegt kein INTERRUPT vor, so beginnt jeder Maschinenzyklus mit $T_1$ und $T_2$. An $T_2$ schließt sich der Zustand $T_3$ an, wenn das Signal READY „High" ist. Ist der Pegel dieses Signals „Low", so folgt nach $T_2$ der Zustand WAIT ($T_W$). Dieser Zustand bleibt so lange erhalten, bis der Pegel des Signals READY wieder „High" wird. Nach WAIT kommt der Zustand $T_3$. Nach $T_3$ kann der Mikroprozessor bei Vorliegen eines HALT-Befehls in den Stoppzustand gehen. Liegt kein HALT-Befehl vor, so schließt sich nach $T_3$ entweder $T_4$ oder, wenn der betreffende Maschinenzyklus zu Ende ist, wieder $T_1$ an. Auf $T_4$ folgt entweder $T_5$ oder, bei Maschinenzyklusende, $T_1$. Nach jedem Ende eines Maschinenzyklus wird ein internes IFF (INTERRUPT-Flip-Flop) abgefragt, ob der betreffende Zyklus ein Programmunterbrechungszyklus ist. Außerdem wird am Ende jedes Befehls das Signal

INTERRUPT (INT) abgefragt, ob es den logischen Pegel „High" hat. Weist es den Pegel „High" auf, so schaltet sich das IFF ein. Ist der laufende Zyklus ein INTERRUPT-Zyklus, oder liegt am Befehlsende eine Programmunterbrechung vor (INT-Signal = „High"), so wird nicht der Zustand $T_1$, sondern der Zustand $T_{1I}$ eingestellt. Auf $T_{1I}$ folgt sofort der Zustand $T_2$.
Aus dem Stoppzustand kommt der Mikroprozessor nur durch eine Programmunterbrechung heraus.

## 2.4. Befehlsabarbeitung

Befehlszyklus
Zur Abarbeitung eines Befehls benötigt der Mikroprozessor mehrere Maschinenzyklen (maximal 3). Zur Unterscheidung der einzelnen Maschinenzyklen dienen Bit 6 und 7 des höherwertigen Adreßbytes.. Man unterscheidet folgende Zyklusarten:

| Bit 6 | Bit 7 | Zyklus | Beschreibung |
|---|---|---|---|
| 0 | 0 | PCI | Speicher-Lese-Zyklus für 1. Byte des Befehls |
| 0 | 1 | PCR | Speicher-Lese-Zyklus für weitere Bytes des Befehls oder Datenbytes |
| 1 | 0 | PCC | Ein- und Ausgabe-Zyklus |
| 1 | 1 | PCW | Schreibzyklus. |

In den einzelnen Zyklen eines Befehls werden in der Regel folgende Operationen ausgeführt:

Zyklus 1

| | | |
|---|---|---|
| Operationscode holen | $T_1$ | NWT-Adresse über DB ausgeben, |
| | $T_2$ | HWT-Adresse über DB ausgeben, |
| | $T_3$ | Operationscode über DB nach BR holen, |
| | $T_4/T_5$ | Ausführung des Befehls, wenn kein weiterer Speicherzugriff erforderlich ist. |

Zyklus 2

| | | |
|---|---|---|
| 2. Speicherzugriff | $T_1$ | NWT-Adresse über DB ausgeben, |
| | $T_2$ | HWT-Adresse über DB ausgeben, |
| | $T_3$ | Speicherzugriff (Byte holen oder abspeichern). |

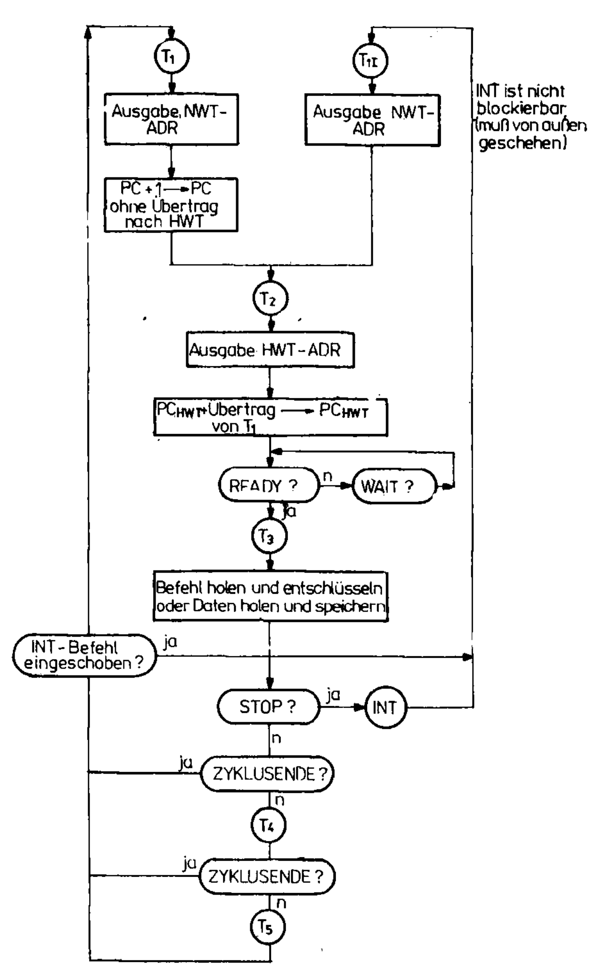



Bild 2.9 Flußdiagramm der einzelnen Schritte der Befehlsabarbeitung im *U 808 D*

Zyklus 3
3. Speicherzugriff wie Zyklus 2
Bild 2.9 zeigt die einzelnen Schritte der Befehlsabarbeitung als Flußdiagramm.
Zur Berechnung der Ausführungszeit eines Befehls geht man von der Grundfrequenz des Taktgenerators aus. Beträgt diese wie in den meisten Fällen 500 kHz (1 Takt = 2 μs), so ist ein Zeitzustand (2 Takte) 4 μs lang. Da ein Zyklus 3 bis 5 Zeitzustände und 1 Befehl 1 bis 3 Zyklen lang sein kann, liegt die Ausführungszeit eines Befehls zwischen 12 und 44 μs.

## 2.5. Befehlsliste *U 808 D*

### 2.5.1. Erläuterung der in der Befehlsliste verwendeten Abkürzungen

Die Abkürzung S sagt aus, daß aus dem Register der Nummer S ein Datenwort geholt wird (S: Source, Quelle).
Die Abkürzung D sagt aus, daß in das Register der Nummer D ein Datenwort gespeichert wird (D: Destination, Senke).
Die Registernummer wird innerhalb des Befehlscodes als Oktalzahl dargestellt.
Den einzelnen Registern sind im Befehlscode folgende Registernummern zugeordnet (s. a. Registerstruktur *U 808 D*, Bild 2.1):

| Register | Registernummer |
|---|---|
| A | 000 = 0 |
| B | 001 = 1 |
| C | 010 = 2 |
| D | 011 = 3 |
| E | 100 = 4 |
| H | 101 = 5 |
| L | 110 = 6 |
| M | 111 = 7 |

M (Memory) bedeutet: Inhalt einer Speicherzelle, deren Adresse im Registerpaar HL steht.

Weitere Abkürzungen sind (s. Bild 2.1)
SP: Stackpointer, Stackzähler
PC: Programm-Counter, Befehlszähler
BR: Befehlsregister

Als Ergebnis einer Rechenoperation werden 4 Flags gestellt. Das sind die Flags Z, C, P, S.

Die Abkürzungen für allgemeine Größen und Register lauten:

r – 8-Bit-Register oder 1 Speicherzelle;
s – 8-Bit-Quellregister oder 1 Speicherplatz;
d – 8-Bit-Bestimmungsregister oder 1 Speicherplatz;
*n* – 8-Bit-Zahl;
*nn* – 16-Bit-Zahl;
Index L – der niederwertige Teil von 16 Bit;
Index H – der höherwertige Teil von 16 Bit.

Es gelten folgende Schreibregeln:
Steht ein Registername allein, z. B. A. so heißt das:
Inhalt von Register A.
Steht ⟨HL⟩, so heißt das:
Inhalt der Speicherzelle, deren Adresse in HL steht.
Für die einzelnen Befehle werden Abkürzungen (*Mnemoniks*) verwendet. Bei der Flagdarstellung bedeutet:

. – Flag wird nicht geändert.
↕ – Flag wird durch den Befehl entsprechend seiner Bedeutung geändert.
0 – Flag wird rückgesetzt.
1 – Flag wird gesetzt.

## 2.5.2. Beschreibung der Befehle des Mikroprozessors *U 808 D*

### 2.5.2.1. Transportoperationen

– Einzelworttransfer

| MOV r, s | s → r | C S Z P |
|---|---|---|
| | | . . . . |

Der Inhalt von Register s wird in ein Register r gebracht.
r kann sein: – Register A, B, C, D, E, H, L;
s kann sein: – Register A, B, C, D, E, H, L;
– M, d. h. eine Speicherzelle, deren Adresse in HL steht.

| MVI r, n | n → r | C S Z P |
|---|---|---|
| | | . . . . |

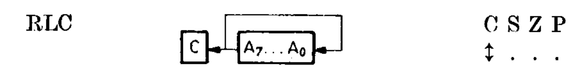
Die Zahl n wird in ein Register r gebracht.

r kann sein: – Register A, B, C, D, E, H, L;
– M, d. h. eine Speicherzelle, deren Adresse in HL steht.

#### 2.5.2.2. Rechen- und logische Operationen mit einem Operand

– Verschiebebefehle

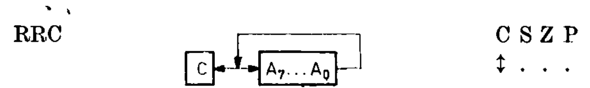
RLC




C S Z P
↕ . . .

Der Inhalt des A-Registers wird um eine Stelle nach links verschoben. Das aus Bit $A_7$ heraustretende Bit wird in das C-Bit und in Bit $A_0$ eingetragen.

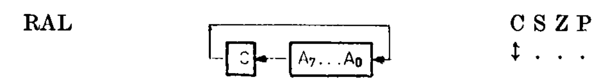
RRC




C S Z P
↕ . . .

Der Inhalt des A-Registers wird um eine Stelle nach rechts verschoben. Das aus Bit $A_0$ heraustretende Bit wird in das C-Bit und in Bit $A_7$ eingetragen.

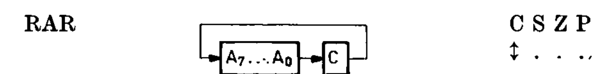
RAL




C S Z P
↕ . . .

Der Inhalt des A-Registers wird zusammen mit dem C-Bit um eine Stelle nach links verschoben. Das aus Bit $A_7$ kommende Bit wird in das C-Bit und das C-Bit in Bit $A_0$ eingetragen.

RAR




C S Z P
↕ . . .

Der Inhalt des Registers A wird zusammen mit dem C-Bit um eine Stelle nach rechts verschoben. Bit $A_0$ kommt in das C-Bit und das C-Bit nach Bit $A_7$.

– 8-Bit-Befehle

INR r $r + 1 \rightarrow r$

C S Z P
. ↕ ↕ ↕

Der Inhalt des Registers r wird um 1 erhöht.

r kann sein: B, C, D, E, H, L.

| | | |
|---|---|---|
| DCR r | $r - 1 \rightarrow r$ | C S Z P<br>↕ ↕ ↕ |

Der Inhalt des Registers r wird um 1 erniedrigt.

r kann sein: B, C, D, E, H, L.

### 2.5.2.3. Rechen- und logische Operationen mit zwei Operanden

| | | |
|---|---|---|
| ADD r | $A + r \rightarrow A$ | |
| ADC r | $A + r + C \rightarrow A$ | |
| SUB r | $A - r \rightarrow A$ | |
| SBB r | $A - r - C \rightarrow A$ | |
| ANA r | $A \wedge r \rightarrow A$ | C S Z P<br>↕ ↕ ↕ ↕ |
| XRA r | $A \oplus r \rightarrow A$ | |
| ORA r | $A \vee r \rightarrow A$ | |
| CMP r | Vergleich A mit r<br>Setzen der Flags nach $A - r$<br>$1 \rightarrow Z$, wenn $A = r$<br>$0 \rightarrow Z$, wenn $A \neq r$<br>$1 \rightarrow C$, wenn $A < r$<br>$0 \rightarrow C$, wenn $A \geqq r$ | C S Z P<br>↕ ↕ ↕ ↕ |

Bei den Rechen- und logischen Operationen mit 2 Operanden wird der Inhalt des A-Registers und der Inhalt des Registers r durch die Operation verknüpft. Das Ergebnis kommt in das A-Register. Bei ADC und SBB wird außerdem das C-Bit zur niederwertigsten Stelle addiert (ADC) bzw. von der niederwertigsten Stelle subtrahiert (SBB). Bei der Vergleichsoperation (CMP) bleiben die Inhalte der Register A und r erhalten. Es werden lediglich die Flags gestellt.

r kann sein:
- Register A, B, C, D, E, H, L;
- M, d. h. eine Speicherzelle, deren Adresse in HL steht.

*Befehle mit Direktoperand*

| | | | Flags |
|---|---|---|---|
| ADI n | $A + n \rightarrow A$ | | |
| ACI n | $A + n + C \rightarrow A$ | | |
| SUI n | $A - n \rightarrow A$ | | |
| SBI n | $A - n - C \rightarrow A$ | | C S Z P |
| ANI n | $A \wedge n \rightarrow A$ | | ↕ ↕ ↕ ↕ |
| XRI n | $A \oplus n \rightarrow A$ | | |
| ORI n | $A \vee n \rightarrow A$ | | |
| CPI n | Vergleich A mit n | | C S Z P |
| | Setzen der Flags nach A − n | | ↕ ↕ ↕ ↕ |
| | $1 \rightarrow Z$, wenn $A = n$ | | |
| | $0 \rightarrow Z$, wenn $A \neq n$ | | |
| | $1 \rightarrow C$, wenn $A < r$ | | |
| | $0 \rightarrow C$, wenn $A \geqq r$ | | |

Bei den Befehlen mit Direktoperand wird der Inhalt des Registers A und die Zahl n (8 Bit) durch die Operation verknüpft. Das Ergebnis kommt in das Register A. Bei den Befehlen ACI und SBI wird außerdem das C-Bit zur niederwertigsten Stelle addiert (ACI) bzw. von der niederwertigsten Stelle subtrahiert (SBI). Bei der Vergleichsoperation (CPI) bleibt der Inhalt des A-Registers erhalten. Es werden lediglich die Flags gestellt.

### 2.5.2.4. Sprungbefehle

*Unbedingter Sprung*

| | | Flags |
|---|---|---|
| JMP nn | $nn \rightarrow PC$ | C S Z P |
| | | • • • • |

Die Adresse nn wird in den Befehlszähler PC gebracht. Der nächste abzuarbeitende Befehl ist damit der Befehl aus Zelle nn. Man sagt, der Rechner führt einen Sprung nach Zelle nn aus. nn darf für den Baustein *U 808 D* nur 14 Bit lang sein.

*Bedingte Sprünge*

| | | Flags |
|---|---|---|
| JNC nn | $nn \rightarrow PC$, wenn C-Flag = 0 | |
| JNZ nn | $nn \rightarrow PC$, wenn Z-Flag = 0 | |
| JP nn | $nn \rightarrow PC$, wenn S-Flag = 0 | |
| JPO nn | $nn \rightarrow PC$, wenn P-Flag = 0 | C S Z P |
| JC nn | $nn \rightarrow PC$, wenn C-Flag = 1 | . . . . |
| JZ nn | $nn \rightarrow PC$, wenn Z-Flag = 1 | |
| JM nn | $nn \rightarrow PC$, wenn S-Flag = 1 | |
| JPE nn | $nn \rightarrow PC$, wenn P-Flag = 1 | |

Ein bedingter Sprung wird so wie ein Sprungbefehl ausgeführt, d. h., die Adresse nn wird in den Befehlszähler PC gebracht, wenn die zugehörige Bedingung erfüllt ist; sonst verhält sich der bedingte Sprungbefehl wie ein Leerbefehl.

### 2.5.2.5. Unterprogrammbefehle

*Unterprogrammrufe*

*Unbedingte Unterprogrammrufe*

| | | |
|---|---|---|
| CALL nn | SP + 1 → SP | C S Z P |
| | PC → ⟨SP⟩ | . . . . |
| | nn → PC | |

Der CALL-Befehl realisiert einen Sprung in ein Unterprogramm, dessen Startadresse nn ist. Dabei wird der aktuelle Befehlszählerstand (Adresse des Operationscodes des nächsten Befehls) in die Stack-Zelle gespeichert, deren Adresse in SP steht. Die Stack-Adresse (Inhalt von SP) erhöht sich vor der Abspeicherung des Befehlszählerinhalts um 1.

*Bedingte Unterprogrammrufe*

| | | |
|---|---|---|
| CNC nn | Ausführung von CALL nn, wenn C-Flag = 0 | |
| CNZ nn | Ausführung von CALL nn, wenn Z-Flag = 0 | |
| CP nn | Ausführung von CALL nn, wenn S-Flag = 0 | |
| CPO nn | Ausführung von CALL nn, wenn P-Flag = 0 | C S Z P |
| CC nn | Ausführung von CALL nn, wenn C-Flag = 1 | . . . . |
| CZ nn | Ausführung von CALL nn, wenn Z-Flag = 1 | |
| CM nn | Ausführung von CALL nn, wenn S-Flag = 1 | |
| CPE nn | Ausführung von CALL nn, wenn P-Flag = 1 | |

Ein bedingter Unterprogrammruf wird so wie ein CALL-Befehl ausgeführt, wenn die zugehörige Bedingung erfüllt ist. Bei Nichterfüllung der Bedingung verhält sich der bedingte Unterprogrammruf wie ein Leerbefehl, d. h., es wird zum nächsten Befehl übergegangen.

*Restartbefehl*

| | | |
|---|---|---|
| RST n | SP + 1 → SP | C S Z P |
| | PC → ⟨SP⟩ | . . . . |
| | n → PC | |

Der Restartbefehl RST n entspricht dem Unterprogrammruf

CALL n. Dabei darf n nur eine der folgenden Hexadezimalzahlen sein:
n = OH, 8H, 10H, 18H, 20H, 28H, 30H, 38H.
H deutet darauf hin, daß die Zahl als Hexadezimalzahl zu lesen ist. Der Unterschied zum eigentlichen CALL-Befehl liegt nur in der Codierung. Der RST-Befehl ist ein 1-Byte-Befehl, wobei die Zahl n bereits dazugehört. Der Befehl CALL nn ist ein 3-Byte-Befehl. Davon benötigt die Adresse nn 2 Byte. Der RST-Befehl wird für die INTERRUPT-Behandlung gebraucht.

*Rückkehrbefehle*
*unbedingter Rückkehrbefehl*

| | | C S Z P |
|---|---|---|
| RET | ⟨SP⟩ → PC | |
| | SP - 1 → SP | . . . . |

Der RET-Befehl realisiert den Rücksprung aus einem Unterprogramm. Durch ihn wird die zuletzt in den STACK gespeicherte Adresse in den Befehlszähler ⟨PC⟩ gebracht. Die dazugehörige STACK-Adresse steht im STACK-Zähler ⟨SP⟩. Nachdem die Adresse in den Befehlszähler gebracht ist, wird der Inhalt des STACK-Zählers ⟨SP⟩ um 1 erniedrigt.

*Bedingte Rückkehrbefehle*

| | | C S Z P |
|---|---|---|
| RNC | Ausführung von RET, wenn C-Flag = 0 | |
| RNC | Ausführung von RET, wenn Z-Flag = 0 | |
| RP | Ausführung von RET, wenn S-Flag = 0 | |
| RPO | Ausführung von RET, wenn P-Flag = 0 | |
| RC | Ausführung von RET, wenn C-Flag = 1 | . . . . |
| RZ | Ausführung von RET, wenn Z-Flag = 1 | |
| RM | Ausführung von RET, wenn S-Flag = 1 | |
| RPE | Ausführung von RET, wenn P-Flag = 1 | |

Ein bedingter Rückkehrbefehl wird so wie ein RET-Befehl ausgeführt, wenn die zugehörige Bedingung erfüllt ist, sonst verhält sich der bedingte Rückkehrbefehl wie ein Leerbefehl.

### 2.5.2.6. Eingabe- und Ausgabebefehle

*Eingabebefehl für Einzelwort*

| | C S Z P |
|---|---|
| IN n (n = Geräteadresse) | |
| | . . . . |

Innerhalb eines Eingabezyklus wird ein auf dem Datenbus vorhandenes Byte in das Register A gebracht. Die Geräteadresse n wird vorher auf den Datenbus DB gegeben. Sie kann eine Zahl zwischen 0 und 7 sein. Zum Schluß werden die Inhalte der Flags auf den Datenbus DB ausgegeben.

| Ablauf: | Zustand $T_1$ | Ausgabe A → DB |
|---|---|---|
| | Zustand $T_2$ | Ausgabe BR → DB |
| | | BR = 0100 MMM1 |
| | | MMM ≙ duale Geräteadresse |
| | Zustand $T_3$ | Eingabe DB → Zwischenspeicher b |
| | Zustand $T_4$ | Ausgabe Flags C P Z S |
| | | C = D3, P = D2, Z = D1, S = D0 |
| | Zustand $T_5$ | b → A |

*Ausgabebefehl für Einzelwort*

OUT n (n = Geräteadresse)

Innerhalb eines Ausgabezyklus wird der Inhalt des Registers A auf den Datenbus gegeben. Daran schließt sich die Ausgabe der Geräteadresse an. Die Geräteadresse kann eine Zahl zwischen 8 und 31 sein, d. h., es lassen sich 24 Ausgabetore anschließen.

| Ablauf: | Zustand $T_1$ | Ausgabe A → DB |
|---|---|---|
| | Zustand $T_2$ | Ausgabe BR → DB |
| | | BR = 01MMMMM1 |
| | MMMMM | = duale Geräteadresse (nur 8 bis 31) |
| | Zustand $T_3$ | — |
| | Zustand $T_4$ | — |

### 2.5.2.7. Steuerbefehle

*Haltbefehl*

HLT C S Z P

. . . .

Nach Entschlüsselung des Befehls HLT geht der Prozessor in den Stoppzustand. Diesen STOP-Zustand kann er nur durch ein Programmunterbrechungssignal (INTERRUPT) verlassen. Der CPU-Status wird nicht verändert.

*Leerbefehl*

NOP C S Z P

. . . .

Einen direkten Leerbefehl gibt es nicht. Er läßt sich aber durch jeden Befehl MOV r, s realisieren, wenn $r = s$ ist. $r = s$ kann sein: Register A, B, C, D, E, H, L.

## 2.6. INTERRUPT

INTERRUPT bedeutet Unterbrechung des gerade laufenden Programms und Übergang zu einem anderen Programm, das durch das unterbrechende Signal bestimmt wird.
Der Prozessor *U 808 D* hat einen INTERRUPT-Eingang (INT), der nicht maskierbar ist. Liegt an diesem Eingang das Signal H, so wird nach Abarbeitung des gerade laufenden Befehls statt des nächsten $T_1$-Zustands ein Zustand $T_{II}$ durchlaufen.
Während $T_{II}$ erhöht sich der Befehlszähler ⟨PC⟩ nicht. Für die Ausgänge $S_0S_1S_2$ gilt $\overline{S_0}S_1S_2 = 1$. Während $T_{II}$ und $T_2$ wird die Ausgabe des Inhalts von PC auf dem Datenbus wie bei $T_1$ und $T_2$ realisiert. Mit dem nachfolgenden $T_3$ wird ein 8-Bit-Wort, ausgesendet vom unterbrechenden Gerät, über den Datenbus eingelesen und als Befehl interpretiert. Ist der eingelesene Befehl ein Mehrbytebefehl (JMP, CALL), so durchläuft der Prozessor während aller zum Befehl gehörenden Zyklen statt $T_1$ $T_{II}$. Dadurch werden die zum Befehl gehörenden weiteren Byte auch über den Datenbus eingelesen, ohne daß der Befehlszähler (PC) erhöht wird. Bild 2.10 zeigt das Blockdiagramm und Bild 2.11 das Taktdiagramm eines INTERRUPT-Zyklus.

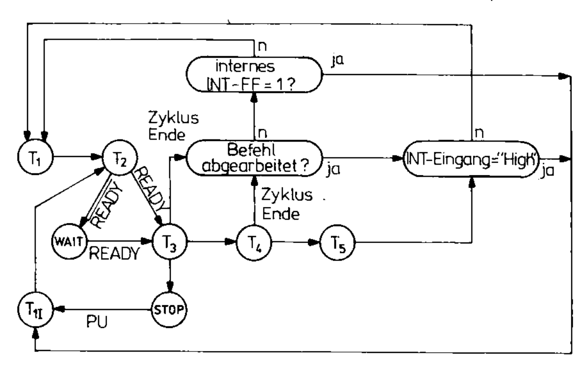



Bild 2.10 Blockdiagramm eines Interruptzyklus des Bausteins *U 808 D*

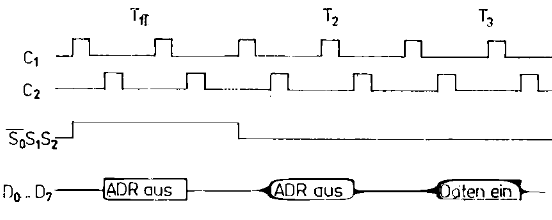



Bild 2.11 Taktdiagramm eines Interruptzyklus im *U 808 D*

Nach $T_{II}$ wird im Zustand $T_3$ durch die externe Logik ein sogenannter INTERRUPT-Befehl über den Datenbus eingegeben. Dieser Befehl löst die Unterbrechung des laufenden Programms und den Sprung in das Bedienprogramm aus. Dazu wird meistens der Befehl RST benutzt. Es kann aber auch ein JMP- oder ein CALL-Befehl sein.

## 2.7. Starten des Prozessors *U 808 D*

Nach dem Einschalten der Spannungen (erst + 5 V, dann − 9 V) werden in den folgenden 8 Zuständen die Register gelöscht (Notizspeicher, STACK und PC). Während dieser 8 Zustände muß das Signal INT L-Pegel haben (16 Takte).
Anschließend geht der Prozessor in den Stoppzustand. Ein Verlassen des Stoppzustands ist nur durch INTERRUPT möglich. Dabei gibt es folgende Varianten:

*Variante 1*
Der bei INTERRUPT während $T_3$ eingelesene Befehl ist ein NOP und kommt aus Zelle 0 des Speichers. Da der Befehlszähler nach Ausführung des Befehls noch auf 0 steht, wird dieser Befehl ein zweites Mal in den Prozessor geholt und abgearbeitet.
Befehlsfolge für Variante 1: 0 NOP
0 NOP
1 Befehl 1
2 Befehl 2

*Variante 2*
Hier steht in der 1. Zelle des Speichers ein RST-Befehl (Code

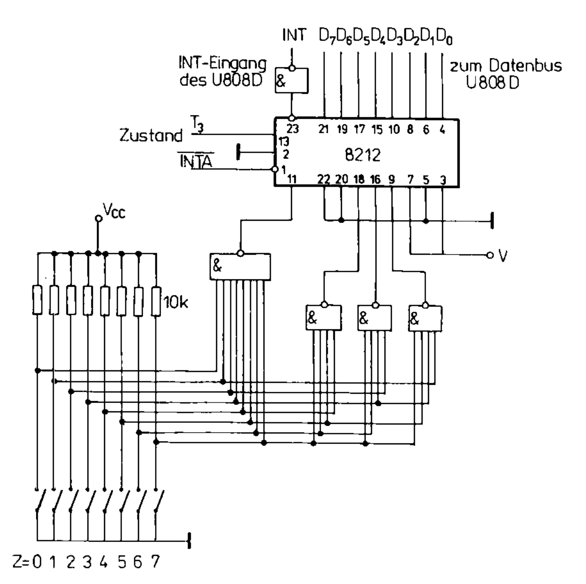



Bild 2.12 Bildung eines Befehls RST Z (Code OZ5) zur Interruptsteuerung im *U 808 D*

OZ5)[1]) zu einer Zelle, in der das Startprogramm beginnt. Hierbei kommt nur eine der Startadressen 8, 16, 32, 40, 48, 56 in Frage.

| Befehlsfolge für Variante 2: | 0 | RST |
|---|---|---|
| | Startadresse | Befehl 1 |
| | Startadresse + 1 | Befehl 2 |

*Variante 3*

Der für INTERRUPT benötigte Befehl (RST, JMP oder CALL) kommt nicht aus dem Speicher, sondern wird durch eine externe Logik realisiert.

Bild 2.12 zeigt eine Möglichkeit zur externen Bildung eines RST-Befehls.

[1]) Der RST-Befehl ist ein Einbytebefehl. Die Oktalziffer Z steht innerhalb des Bytes (00 $\underbrace{\text{XXX}}_{\text{Z}}$ 101). Die Startadresse ist 8Z.

Z ist eine Oktalziffer zwischen 0 und 7.

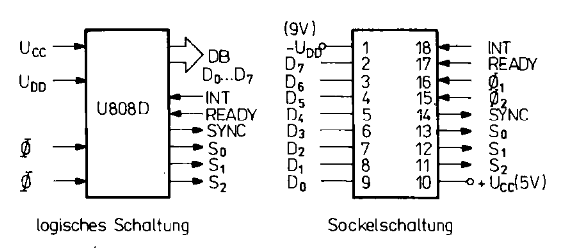



Bild 2.13 Anschlußspannungen und Anschlußsignale des Schaltkreises *U 808 D*

## 2.8. Beschreibung der Anschlüsse des Bausteins *U 808 D*

Bild 2.13 zeigt die Anschlußspannungen und Anschlußsignale des Schaltkreises *U 808 D.*
$U_{CC}$ und $U_{DD}$ sind anzulegende Betriebsspannungen mit folgenden Werten:

| | Minimal | Maximal | Maximale Stromaufnahme |
|---|---|---|---|
| $U_{CC}$ | 4,75 V | 5,25 V | $I_{CC} < 60$ mA |
| $U_{DD}$ | −9,45 V | −8,55 V | $I_{DD} < 60$ mA |

Für die Logiksignale gelten folgende Daten:

| | | Minimal | Maximal |
|---|---|---|---|
| Eingang: | $U_{eL}$ | | $U_{CC} - 4{,}35$ V |
| | $U_{eH}$ | $U_{CC} - 1{,}5$ V | $U_{CC} + 0{,}3$ V |
| Ausgang: | $U_{aL}$ | | 0,4 V bei $I_{aL} = 0{,}4$ mA |
| | $U_{aH}$ | 2,4 V bei $I_{aH} = -0{,}2$ mA | |

$D_0$ bis $D_7$ Bidirektionaler Datenbus (aktiv „High“)
Er wird verwendet für die Ein- und Ausgabe von Daten- und Adreßworten.

INT Eingabesignal (aktiv „High“)
INT führt zu einer Programmunterbrechung im Prozessor. Es wird am Ende jedes Befehls angenommen.

READY Eingabesignal (aktiv „High“)
Ist READY = L, dann wird der Prozessor veranlaßt, nach $T_2$ in den Wartezustand $T_W$ zu

gehen. Solange READY = L ist, führt der Prozessor Wartezyklen aus.

SYNC — Ausgabesignal (aktiv „High“)
SYNC sagt aus, daß sich der Prozessor im 1. $\Phi_1$-Takt eines Zustands befindet.

$S_0$, $S_1$, $S_2$ — Statussignale, die angeben, in welchem Zustand sich entsprechend Tabelle 2.1. der Prozessor befindet.

$\Phi_1$, $\Phi_2$ — Steuertakte — TTL-kompatibel (Taktfolge nach Bild 2.5).

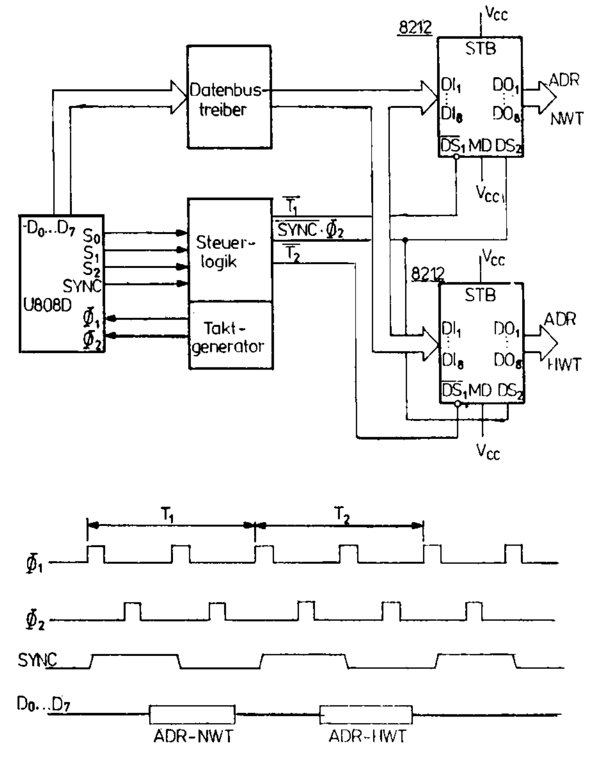



**Bild 2.14** **Bildung der Speicheradresse im Baustein *U 808 D***

## 2.9. Anschluß externer Schaltkreise an den Prozessor *U 808 D*

*Bereitstellung der Adresse*

Wie in Abschnitt 2.4. behandelt, sendet der Prozessor *U 808 D* zum Zeitpunkt $T_1$ den niederwertigen Teil und zum Zeitpunkt $T_2$ den höherwertigen Teil der Adresse über den Datenbus DB aus. Aus dem Taktdiagramm in Bild 2.14 ist ersichtlich, daß die Adreßwerte immer im 2. Teil eines Zustands, d. h. während $\overline{SYNC}$ am Datenbus anliegen. Die Adreßteile müssen in einem Register zwischengespeichert werden. Bildet man ein Signal

$\overline{SYNCA} = \overline{SYNC} \cdot \Phi_2$,

so ist das Torungssignal für den niederwertigen Adreßteil $ADR_L$,

$ADR_L = T_1 \cdot \overline{SYNCA}$,

wobei $T_1 = \bar{S}_0 S_1 \bar{S}_2$ ist, und das Torungssignal für den höherwertigen Adreßteil $ADR_H$

$ADR_H = T_2 \cdot \overline{SYNCA}$,

wobei $T_2 = \bar{S}_0 \bar{S}_1 S_2$ ist.

Bild 2.14 zeigt die Ausgabe der Adresse in 2 Registerbausteine *8212*. Die Adreßbits $A_{14}$ und $A_{15}$ dienen nicht als Adresse, sondern zur Unterscheidung zwischen den einzelnen Zyklusarten gemäß Abschnitt 2.4.

*Eingabe von Daten in den Prozessor*

Aus dem Taktdiagramm in Bild 2.15 ist ersichtlich, daß die Eingabe über den Datenbus zum Zeitpunkt $T_3$ während des Signals SYNC vor sich geht.

Dateneingabe: $DBIN = T_3 \cdot SYNC$.

Durch Bit 6 und Bit 7 des höherwertigen Adreßbytes wird zusätzlich festgelegt, woher das einzugebende Byte kommt. Bit 6 des höherwertigen Adreßbytes sei mit $A_{14}$ und Bit 7 mit $A_{15}$ bezeichnet (s. Bild 2.14).

Nach Abschnitt 2.4. gilt:

Ist $A_{14} = 0$ und $A_{15} = 0$, so handelt es sich um eine Eingabe vom Speicher (PCI-Zyklus).

Ist $A_{14} = 0$ und $A_{15} = 1$, dann handelt es sich auch um eine Eingabe vom Speicher (PCR-Zyklus).

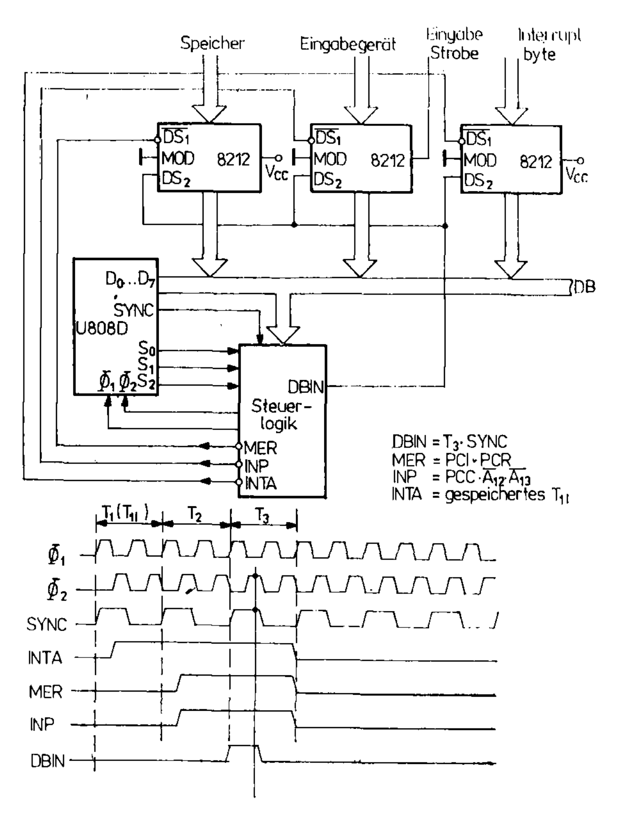



**Bild 2.15** **Blockdiagramm zur Realisierung der Eingabe eines Bytes vom Speicher, vom Eingabegerät oder als Interruptbyte in den Prozessor *U 808 D***

Ist $A_{14} = 1$ und $A_{15} = 0$, so handelt es sich um einen Eingabe-/Ausgabe-Zyklus (PCC-Zyklus). Bei der Eingabe muß zusätzlich $A_{12}$ und $A_{13}$ (Bit 4 und Bit 5 des höherwertigen Adreßbytes) 0 sein.

War zum Zeitpunkt $T_1$ ein $T_{11}$-Zyklus (INTERRUPT-Zyklus), so wird während DBIN ein INTERRUPT-Wort eingegeben. Das

$T_{II}$-Signal muß dazu in einem Flip-Flop INTA gespeichert werden, das mit dem Ende von $T_3$ oder mit $T_1$ gelöscht werden kann:

$T_{II} \rightarrow INTA$

Ende $T_3 = T_3 \cdot \overline{SYNCA} \rightarrow \overline{INTA}$

Es gilt also für das Eingabetorungssignal $T_E$:

Eingabe vom Speicher $T_E = DBIN \cdot (PCI \vee PCR)$

$PCI = \overline{A_{14}} \cdot \overline{A_{15}}$

$PCR = \overline{A_{14}} \cdot A_{15}$

Eingabe vom Eingabetor $T_E = DBIN \cdot PCC \cdot \overline{A_{12}} \cdot \overline{A_{13}}$

$P_{CC} = A_{14} \cdot \overline{A_{15}}$

Eingabe INTERRUPT-Wort $T_E = DBIN \cdot INTA$

Bild 2.15 zeigt ein Blockdiagramm zur Realisierung dieser 3 Eingabemöglichkeiten, wobei die Abkürzungen MER — MEMORY READ (Speicher lesen) und INP — INPUT verwendet wurden.

$MER = PCI \vee PCR$

$INP = PCC \cdot \overline{A_{12}} \cdot \overline{A_{13}}$

Für die Realisierung der Eingabetore dienen Registerschaltkreise *8212* (s. Teil 1, Abschnitt 3.4.).

*Ausgabe von Daten*

Eine Ausgabe von Daten über den Datenbus $D_0$ bis $D_7$ des Prozessors *U 808 D* erfolgt zum Zeitpunkt $T_3$, wenn SYNC L-Pegel hat. Im PCW-Zyklus werden die Daten zum Speicher ausgegeben, im PCC-Zyklus zu einem Ausgabegerät. Im PCC-Zyklus liegt Ausgabe vor, wenn Bit 4 oder Bit 5 des höherwertigen Adreßbytes ($A_{12}$ oder $A_{13}$) 1 ist.

Torungssignal für Ausgabe zum Speicher:

$T_3 \cdot \overline{SYNCA} \cdot PCW$ $\overline{SYNCA} = \overline{SYNC} \cdot \Phi_2$

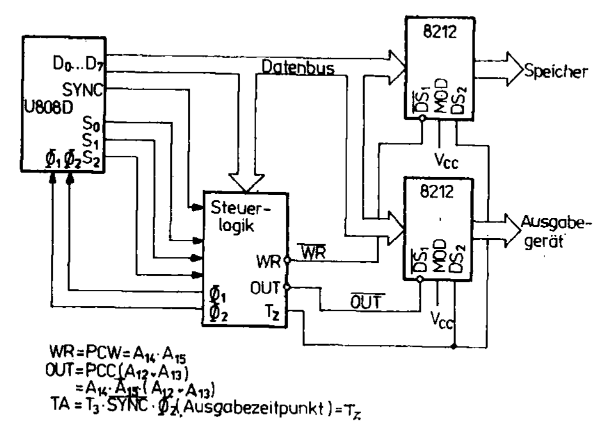
$PCW = A_{14} \cdot A_{15}$

Torungssignal für die Ausgabe zum Ausgabetor:

$T_3 \cdot \overline{SYNCA} \cdot PCC\ (A_{12} \vee A_{13})$ $PCC = A_{14} \cdot \overline{A_{15}}$

Bild 2.16 zeigt das Block- und Taktdiagramm für die Ausgabe zum Speicher und zum Ausgabetor. Für die Realisierung der Tore lassen sich wieder Registerbausteine *8212* verwenden.

Da die Ausgabesignale des Prozessors *U 808 D* nur mit einer Lasteinheit belastbar sind, müssen in den Datenbus Bustreiber geschaltet werden. Dazu eignen sich Bausteine vom Typ *8216* oder andere Bustreiberschaltkreise (s. Teil 1, Abschnitt 3.7.).



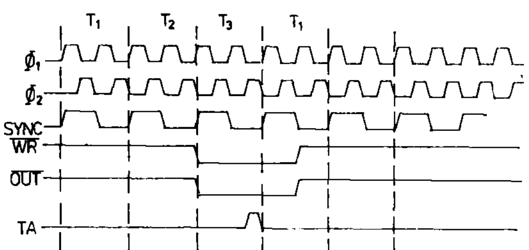




Bild 2.16 Blockdiagramm zur Realisierung der Ausgabe eines Bytes vom Prozessor *U 808 D* zum Speicher und zum Ausgabegerät

# 3. Der Mikroprozessorbaustein *U 880 D*

Der Baustein *U 880* ist gegenüber dem Baustein *U 808 D* weiter vervollkommnet. Die wesentlichen Verbesserungen sind folgende:

- Es ist nur eine Betriebsspannung $V_{cc} = 5$ V notwendig und ebenfalls nur ein Steuertakt $\Phi$ erforderlich.
- Die Steuersignale zur Auswahl und Ansteuerung der externen Bausteine werden im Prozessor schon so weit aufbereitet, daß sie direkt mit den Eingängen und Ausgängen der externen Bausteine verbunden werden können.
- Die Befehlsliste ist wesentlich erweitert; während der Baustein *U 808 D* 48 Basisbefehle verarbeitet, sind es beim *U 880* 158 Befehle. Neu sind dabei Befehle für Doppelwortoperationen, für BCD-Arithmetik, für einen zweiten Registersatz, Blocktransferbefehle und Blocksuchbefehle in Verbindung mit dem Speicher und den Ein- und Ausgabebausteinen, Bitoperationen, Indexoperationen sowie wesentlich erweiterte Verschiebebefehle.
- Die Behandlung von externen INTERRUPT ist durch einen maskierten INTERRUPT sowie durch die Möglichkeit des Aufbaus einer Adreßliste für unterschiedliche INTERRUPT-Routinen erweitert worden.

## 3.1. Registerstruktur des Mikroprozessorbausteins *U 880*

Aus Bild 3.1 ist die Registerstruktur des Bausteins *U 880* zu ersehen. Der Mikroprozessor enthält einen internen 8-Bit-Bus, von dem aus alle Register zu erreichen sind. Von diesem Bus aus werden Daten über den externen Datenbus $D_0$ bis $D_7$ ein- und ausgegeben. Er läßt sich in beiden Richtungen betreiben. Der externe bidirektionale Datenbus ist über Bustreiber mit dem internen 8-Bit-Bus verbunden. An den internen 8-Bit-Bus sind angeschlossen:

- 2 Registersätze zur Zwischenspeicherung der Zahlen im Prozessor, die aus je 8-Bit-Registern bestehen Der 1. Registersatz

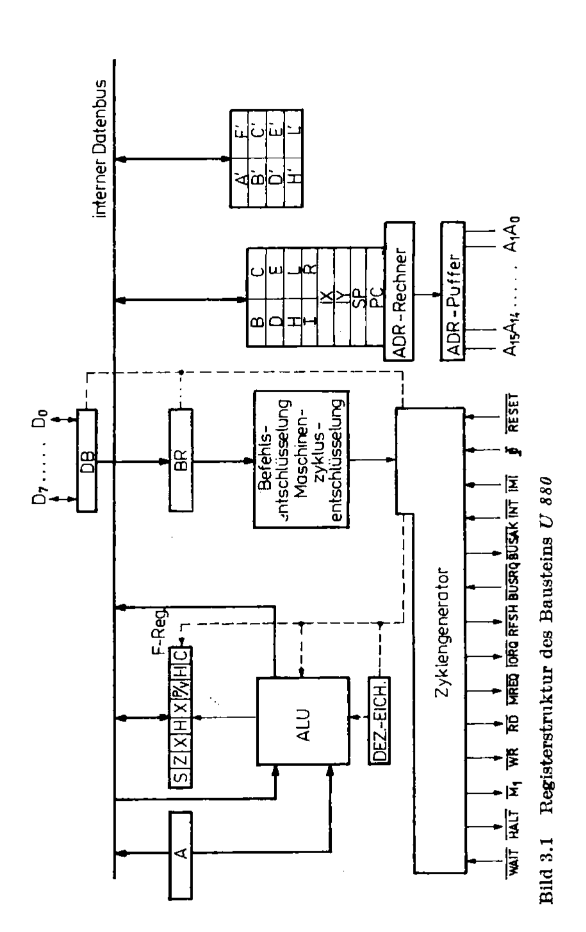



Bild 3.1 Registerstruktur des Bausteins *U 880*

beinhaltet die Register A, F[2]), B, C, D, E, H, L, der 2. Registersatz die Register A′, F′[2]), B′, C′, E′, H′, L′. Durch einen einfachen Austauschbefehl können die Inhalte der Registersätze komplett vertauscht werden. Dadurch ist es möglich, einen bestimmten Programmabschnitt einem der Registersätze zuzuordnen. Wechselt das Programm, dann können die dazugehörigen Registersätze umgetauscht werden.

2) F, F′ sind Flagregister

- Zweckregister I, R, IX, IY, SP, PC.
  Das Register I (8-Bit-INTERRUPT-Adreßregister) enthält den höherwertigen Teil einer Adresse, deren niederwertiger Teil bei einem INTERRUPT von dem entsprechenden Gerät gebildet wird. Die Adresse weist auf eine Speicherzelle, in der die Startadresse des INTERRUPT-Bedienprogramms steht.
  Das Register R (Speicherauffrischregister) enthält eine 7-Bit-Adresse, die in Verbindung mit dem Auffrischsignal RFSH auf den niederwertigen 7 Bit des Adreßbus ausgesendet wird. Diese Adresse wird während der Operationscodeentschlüsselung ausgesendet. Sie dient zum Auffrischen von dynamischen Speichern. Während jedes Operationscodeholzyklus erhöht sich der Inhalt des Registers R um 1.
  Die beiden Register IX und IY (Indexregister) können eine 16-Bit-Basisadresse enthalten. Während der Adressenrechnung wird aus dieser Basisadresse durch Addition einer Adreßzahl die eigentliche Operandenadresse ermittelt.
  Das Register SP (Stackpointer) enthält eine 16-Bit-Adresse, die die Speicherzelle an der Spitze eines Kellerspeichers adressiert. Der Kellerspeicher ist als „last in – first out-Speicher" organisiert (das zuletzt eingeschriebene Wort wird zuerst gelesen).
  Das Register PC (Program-Counter oder Befehlszähler) enthält eine 16-Bit-Adresse, die angibt, aus welcher Speicherzelle der laufende Befehl geholt wird.
- In dem Befehlsregister BR wird der Operationscode des laufenden Befehls gespeichert. Hier kommt es zur Decodierung des Befehls und zur Bildung der Steuersignale für dessen Abarbeitung. Die Steuersignale bestehen aus den Befehls- und den Zeitsignalen. Die Befehlssignale werden durch die Befehlsentschlüsselung und die dazugehörigen Zeitsignale durch die Zeitsteuerung gebildet. Die Zeitsteuerung besteht aus dem Zyklengenerator, der durch den externen Takt $\Phi$ und die Befehlssignale gesteuert wird. Im Zyklengenerator werden auch die Signale zur Steuerung der externen Bausteine gebildet sowie die von den externen Bausteinen kommenden Signale abgetastet.
- Das Flagregister enthält 6 Flip-Flop, die in Abhängigkeit von den einzelnen Befehlen und vom Ergebnis der Befehle gesetzt oder rückgesetzt werden. Die einzelnen Flags haben folgende Bedeutung:

C: Carry-Flag
C ist gleich 1, wenn bei der Addition ein Übertrag in die 8. Stelle auftritt, oder wenn bei der Subtraktion ein Borgen von der 8. Stelle notwendig wird.
N: Subtraktions-Flag
N ist gleich 1, wenn die ausgeführte Operation eine Subtraktion war.
P/V: Parity-Überlauf-Flag (Überlauf = Overlow)
P/V ist gleich 1, bei logischen Operationen, wenn die Anzahl der Einsen im Ergebnis geradzahlig ist, bei Rechenoperationen, wenn ein Überlauf auftritt (Ergebnis größer als die größte darstellbare Zahl).[3]
H: Half-Carry-Flag
H ist gleich 1, wenn es bei der Addition zu einem Übertrag in die 4. Stelle kommt oder wenn bei der Subtraktion ein Borgen von der 4. Stelle notwendig wird.
Z: Zero-Flag
Z ist gleich 1, wenn das Ergebnis 0 ist.
S: Sign-Flag
S ist gleich 1, wenn im Ergebnis das Vorzeichen 1 (negativ) ist.

*Beispiele*

```
                      1 ↖          Übertrag in 4. Stelle
   120 =    0 1 1 1     1 0 0 0    ergibt 1 → H
 + 105 =    0 1 1 1     1 0 0 1
 ---------------------------------
   225 = 0] 1 1 1 1     0 0 0 1
         ↙    ↘
 0 → C     1 → P/V        wegen Überlauf

                                   kein Übertrag in 4. Stelle
                                   ergibt 0 → H
  − 5 =     1 1 1 1     1 0 1 1
  −16 =     1 1 1 1     0 0 0 0
 ---------------------------------
  −21 = 1]  1 1 1 0     1 0 1 1
 1 → C ↙    aber 0 → P/V
```

3) Der Prozessor *U 880* arbeitet mit einem 8-Bit-Zahlwort im Zweierkomplement. Der Stellenwert $2^7$ entspricht dem Vorzeichen. Die größte positive Zahl ist $2^7 - 1$, die negative Zahl mit dem größten Betrag $-2^7$. Der vom Prozessor erfaßte Zahlenbereich umfaßt $-2^7 \leqq Z \leqq 2^7 - 1$.

## 3.2. Befehlsaufbau des Bausteins *U 880*

### 3.2.1. Befehlsstruktur

Ein Befehl besteht aus Operationsteil und Adreßteil (s. Bild 2.3). Zur Darstellung eines Befehls werden 1 bis 4 Byte benötigt. Davon kann der Operationscode 1 bis 3 Byte [4]) und der Adreßteil ebenfalls 1 bis 2 Byte lang sein. Bei den meisten Befehlen ist der Operationscode 1 Byte lang. Bei 2 Byte langen Operationscodes gibt das 1. Byte die Befehlsgruppe an. Durch das 2. Byte und 3. Byte werden spezielle Befehle innerhalb der Gruppe gekennzeichnet. Bild 3.2 zeigt die im Prozessor *U 880* möglichen Befehlsstrukturen.

4) Bei einigen Befehlen mit Indexrechnung ist der Operations code 3 Byte und der Adreßteil 1 Byte lang.

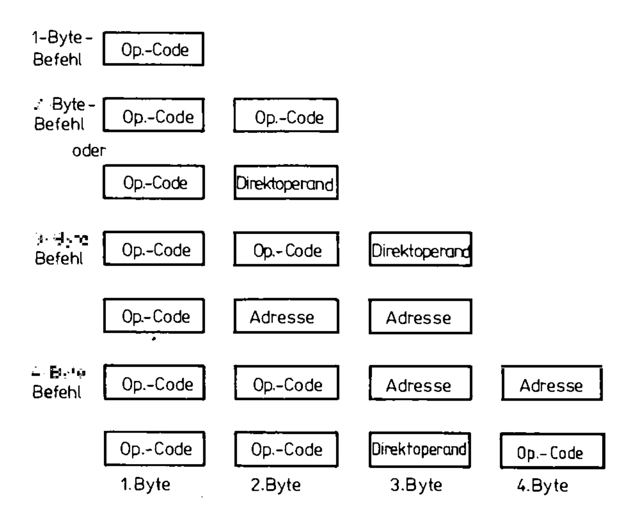



Bild 3.2 Befehlsstrukturen im Prozessor *U 880*

### 3.2.2. Adreßbildung

*Direktoperand*

Der zum Befehl gehörende Operand steht im Anschluß an den Operationscode:

Zelle 1 Operationscode,
Zelle 2 NWT-Operand (niederwertiger Teil des Operanden),
Zelle 3 HWT-Operand (höherwertiger Teil des Operanden).

*Adressierter Operand*

Im Befehl steht die Adresse der Speicherzelle, in der der Operand steht:

Zelle 1 Operationscode,
Zelle 2 NWT-ADR (niederwertiger Teil der Adresse),
Zelle 3 HWT-ADR (höherwertiger Teil der Adresse).

*Relative Adressierung*

Im Befehl steht eine positive oder negative Zahl N.
Der Operand steht um N Zellen nach oder vor dem Befehl.
Zelle 1 Operationscode,
Zelle 2 N (positive oder negative Zahl im Zweierkomplement),
Zelle 3 nächster Befehl;
ADR als Operanden = 3 + N.

*Indirekte Adressierung*

Die Adresse ADR des Operanden steht in einem speziellen Register:
ADR = ⟨Register⟩
Als Register treten die Registerpaare BC, DE, HL sowie die Register SP, IX und IY auf.

*Indexierung*

Die Adresse ADR des Operanden ergibt sich aus der im Befehl angegebenen Zahl N plus dem Inhalt eines Indexregisters.
Zelle 1 Operationscode
Zelle 2 N (positive oder negative Zahl im Zweierkomplement)
ADR = N + ⟨Indexregister⟩

*Registeroperand*

Der Operand steht in einem im Befehl angegebenen Register.

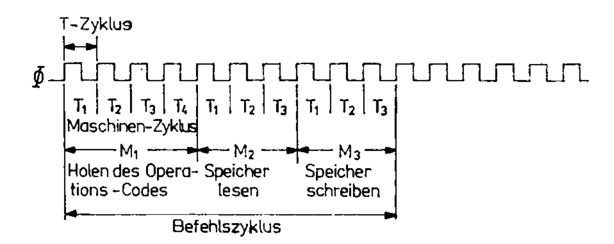



Bild 3.3 Aufbau eines Befehlszyklus im *U 880*

## 3.3. Zeitverhalten

Ein Befehl wird in mehreren Maschinenzyklen abgearbeitet. Es gibt Maschinenzyklen für folgende Funktionen:

- Operationscode holen,
- Speicher lesen oder schreiben.
- Ein- und Ausgabe,
- INTERRUPT,
- DMA-Funktion,
- Ausführung einer HALT-Operation.

Ein Maschinenzyklus unterteilt sich in 3 bis 6 Zustände (T-Zyklen). Ein T-Zyklus entspricht einer Periode des Grundtaktes $\Phi$.

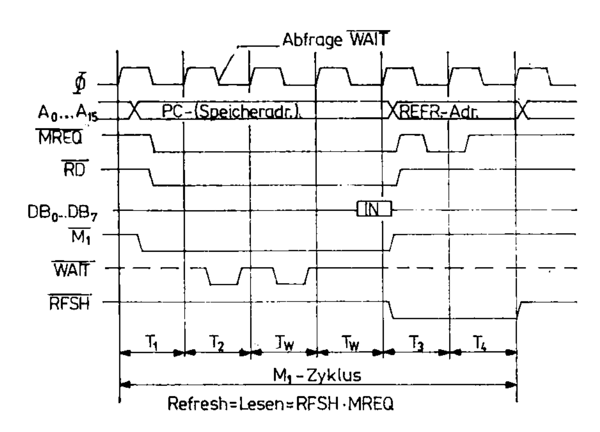



Bild 3.4 Operationscodeholzyklus im *U 880*

Bild 3.3 zeigt ein Beispiel für den Aufbau eines Befehlszyklus (Gesamtzeitraum zur Abarbeitung eines Befehls).

*Operationscode-Holzyklus* (Bild 3.4)
Am Anfang des Zyklus enthält der Befehlszähler die Adresse des Operationscodes. Mit der Rückflanke von $T_1$ wird das Signal $\overline{\text{MREQ}}$ aktiv, gleichzeitig das Signal $\overline{\text{RD}}$. $\overline{\text{MREQ}}$ bedeutet eine Anforderung zum Speicher; $\overline{\text{RD}}$ sagt aus, daß eine Lese-Operation ablaufen soll. Das Einlesen der Daten geschieht mit der Vorderflanke von $\Phi$ während $T_3$. In den Zyklen $T_3$ und $T_4$ wird eine Auffrischadresse für dynamische Speicher an den Adreßbus gelegt. Die Auffrischadresse liegt an den Bitstellen $A_0$ bis $A_6$, während die übrigen Bit 0 sind. Zu dem Zeitpunkt, in dem Auffrischadresse am Adreßbus liegt, ist das Signal $\overline{\text{RFSH}}$ aktiv. Ist zum Zeitpunkt der Rückflanke von $\Phi$ im Zustand von $T_2$ das $\overline{\text{WAIT}}$-Signal aktiv, so wird nach $T_2$ ein Wartezustand eingeschoben. Dieses Einschieben von Wartezuständen wiederholt sich so lange, bis das $\overline{\text{WAIT}}$-Signal inaktiv wird.

*Speicher-Lese- oder -Schreib-Zyklus* (Bild 3.5)
Mit Beginn des Speicher-Lesezyklus (angesteuert durch die Vorderflanke von $\Phi$ im Zustand von $T_1$) wird die Speicheradresse auf den Adreßbus gelegt. Mit der Rückflanke von $\Phi$ (Zustand $T_1$) aktivieren sich die Signale $\overline{\text{MREQ}}$ und $\overline{\text{RD}}$. Das Signal $\overline{\text{MREQ}}$ kann zur Ansteuerung des betreffenden Speichers genommen werden, während $\overline{\text{RD}}$ den Speicher auf Lesen umschaltet. Zum

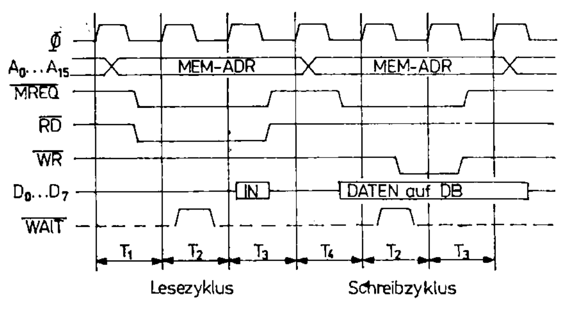



Bild 3.5 Speicher-, Lese- oder Schreibzyklus im *U 880*

Zeitpunkt der Rückflanke von $\Phi$ in $T_2$ tastet der Prozessor das $\overline{\text{WAIT}}$-Signal ab und fügt nach $T_2$ bei aktivem $\overline{\text{WAIT}}$-Signal einen Wartezyklus $T_{wait}$ ein. Während $T_{wait}$ bleiben die Adresse am Adreßbus und die Daten am Datenbus erhalten. Bei der nächsten Rückflanke von $\Phi$ wird die Abfrage von $\overline{\text{WAIT}}$ wiederholt und eventuell ein weiterer Wartezustand eingeschoben. Ist das $\overline{\text{WAIT}}$-Signal nicht mehr aktiv, dann folgt der Zustand $T_3$. Während des Taktes $\Phi$ im Zustand $T_3$ werden die Daten vom Datenbus in den Prozessor übernommen, mit der Rückflanke von $\Phi$ in $T_3$ werden die Signale $\overline{\text{MREQ}}$ und $\overline{\text{RD}}$ wieder abgeschaltet.
Beim Speicher-Schreib-Zyklus wird die Adresse genau wie zum Speicher-Lese-Zyklus mit der Vorderflanke von $\Phi$ in $T_1$ auf den Adreßbus gelegt. Mit der Rückflanke von $\Phi$ in $T_1$ werden die Daten an den Datenbus gelegt. Mit der Rückflanke von $\Phi$ in $T_2$ wird das Signal $\overline{\text{WR}}$ aktiv und gleichzeitig das $\overline{\text{WAIT}}$-Signal abgefragt. $\overline{\text{WR}}$ kann zum Umschalten des Speichers auf Schreiben benutzt werden. Während das Übernehmen der Daten in den Speicher mit $\Phi$ in $T_3$ erfolgen kann, wird mit der Rückflanke von $\Phi$ in $T_3$ $\overline{\text{MREQ}}$ und $\overline{\text{RD}}$ wieder abgeschaltet.

*Ein- und Ausgabe-Zyklus* (Bild 3.6)
Beim Ein- und Ausgabe-Zyklus wird nach $T_2$ automatisch ein Wartezyklus eingefügt, um dem Ein- und Ausgabebaustein zu

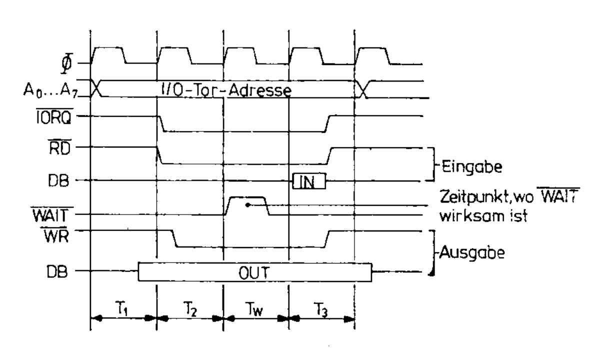



Bild 3.6 Ein- und Ausgabezyklus im *U 880*

ermöglichen, eine Adreßentschlüsselung durchzuführen und im Notfall das $\overline{\mathrm{WAIT}}$-Signal zu setzen. Der Ablauf des Zyklus ähnelt dem des Speicher-Lese- oder -Schreib-Zyklus. Zum Zeitpunkt der Vorderflanke von $\Phi$ in $T_2$ wird das Signal $\overline{\mathrm{IORQ}}$ aktiv. Gleichzeitig aktiviert sich entweder $\overline{\mathrm{RD}}$ oder $\overline{\mathrm{WR}}$, je nachdem, ob es sich um eine Eingabe oder um eine Ausgabe handelt. Bei der Ausgabe erscheinen die Daten auf dem Datenbus bereits während $T_1$, so daß zum Zeitpunkt $\Phi$ in $T_3$ die Daten abgenommen werden können.

*INTERRUPT-Zyklus*

Bild 3.7 zeigt das Zeitdiagramm für den maskierten INTERRUPT-Zyklus. Das Signal INT wird im letzten Zustand eines Befehls abgetastet. Ist es aktiv, dann beginnt mit $T_1$ ein INTERRUPT-Zyklus. Mit der Vorderflanke von $\Phi$ in $T_1$ gelangt die Adresse aus dem Befehlszähler an den Adreßbus. Gleichzeitig wird $\overline{\mathrm{M_1}}$ eingeschaltet. In jedem INTERRUPT-Zyklus werden automatisch 2 WAIT-Zustände eingeschoben, damit die INTERRUPT-Logik genügend Zeit zur Entschlüsselung der Adresse und zur Bereitstellung des INTERRUPT-Vektors hat. Mit der Rückflanke von $\Phi$ im ersten Wartezustand wird zusätzlich das Signal $\overline{\mathrm{IORQ}}$ aktiv. Das gleichzeitige Vorhandensein von $\overline{\mathrm{IORQ}}$ und $\overline{\mathrm{M_1}}$ besagt, daß der INTERRUPT angenommen worden ist. Nach der Rückflanke von $\Phi$ des letzten Wartezustands wird vom Prozessor der Datenbus abgetastet und als INTERRUPT-Vektor übernommen.

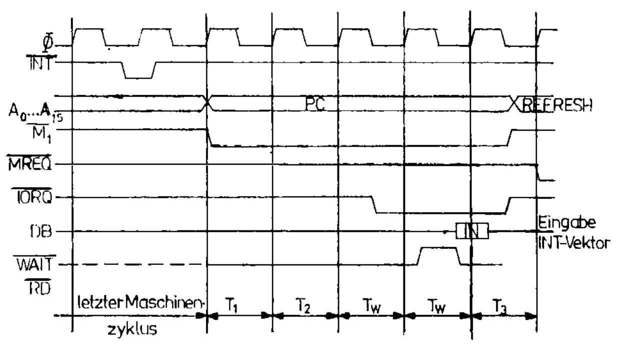



Bild 3.7 Zyklus für den maskierten INTERRUPT

Während $T_3$ und $T_4$ kommt es wie beim Zyklus $M_1$ zur Ausgabe einer Auffrischadresse mit den dazugehörigen Signalen $\overline{MREQ}$ und $\overline{RFSH}$.
In Abhängigkeit vom INTERRUPT-MODE (0, 1, 2) wird der INTERRUPT-Vektor unterschiedlich interpretiert.

*Maskierter INTERRUPT*

MODE 0 Der INTERRUPT-Vektor wird als Befehlscode interpretiert.

MODE 1 Der INTERRUPT-Vektor bleibt unberücksichtigt. Es wird der Befehl CALL 38H gebildet und ausgeführt.

MODE 2 Der INTERRUPT-Vektor wird in Verbindung mit dem I-Register als Adresse interpretiert, die angibt, in welcher Zelle sich die Ansprungsadresse des Bedienungsprogramms befindet. Es wird der Befehl CALL ⟨I-Register, INTERRUPT-Vektor⟩ ausgeführt.

*Nichtmaskierter INTERRUPT*

Es wird der Befehl CALL 66H gebildet und ausgeführt (Taktdiagramm Bild 3.8).

*Haltezyklus* (Bild 3.9)

Nach der Entschlüsselung eines HALT-Befehls führt der Prozessor Leerbefehle (NOP) aus, und zwar so lange, bis ein INTERRUPT erscheint (entweder ein nichtmaskierter oder ein maskierter INTERRUPT, wenn dieser erlaubt ist). Die INTERRUPT-

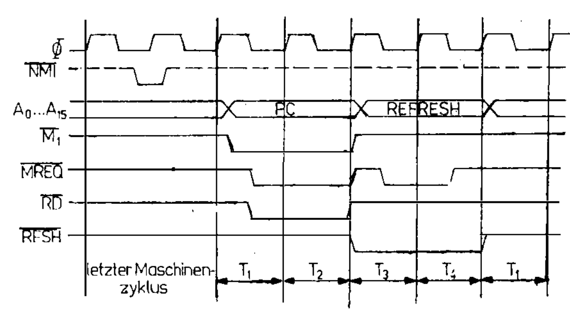



Bild 3.8 Taktdiagramm für den nichtmaskierten INTERRUPT beim *U 880*

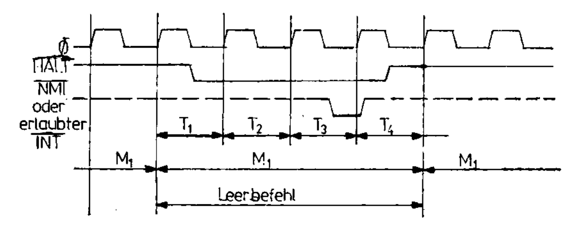



**Bild 3.9** **Haltezyklus im *U 880***

Eingänge werden mit der Vorderflanke von $\Phi$ in $T_4$ abgetastet. Ist zu diesem Zeitpunkt ein INTERRUPT-Eingang aktiv, dann setzt sich mit dem nächsten Takt die Befehlsabarbeitung fort. Es wird ein Sprung an die Stelle ausgeführt, die der entsprechenden INTERRUPT-Behandlung entspricht.

*DMA-Zyklus* (Bild 3.10)

Mit der Vorderflanke von $\Phi$ jedes letzten Taktes eines Maschinenzyklus wird das Signal $\overline{\text{BUSRQ}}$ abgetastet. Ist es zu diesem Zeitpunkt aktiv, so werden mit Beginn des nächsten $T_1$ der Adreßbus, der Datenbus und die Steuersignale $\overline{\text{MREQ}}$, $\overline{\text{RD}}$, $\overline{\text{WR}}$, $\overline{\text{IORQ}}$ und $\overline{\text{RFSH}}$ in den hochohmigen Zustand gesetzt. Gleichzeitig aktiviert sich das Signal $\overline{\text{BUSAK}}$, als Zeichen dafür, daß der hochohmige Zustand erreicht ist. Nun wird in jedem Zustand mit der Vorderflanke von $\Phi$ das Signal $\overline{\text{BUSRQ}}$ abgetastet. Ist es nicht

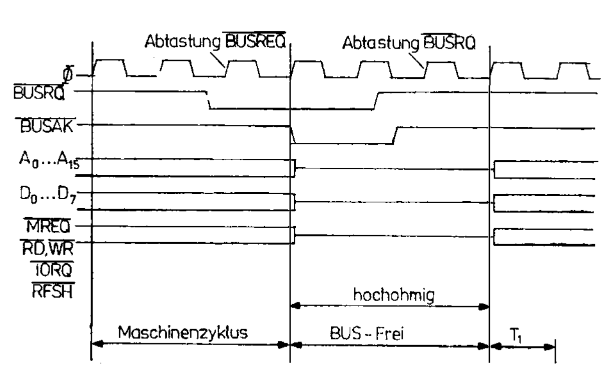



**Bild 3.10** **DMA-Zyklus im *U 880***

mehr aktiv, so wird im nächsten Takt der hochohmige Zustand beendet, und es beginnt ein neuer Maschinenzyklus. Während $\overline{\text{BUSAK}}$ aktiv ist, kann kein INTERRUPT auftreten. Der REFRESH ist unterbrochen.

## 3.4. Befehlsabarbeitung

Während der Abarbeitung eines Befehls werden folgende Arbeitsgänge durchlaufen:
- Befehl holen,
- Befehl entschlüsseln,
- Operand holen,
- Befehl ausführen.

Die einzelnen Arbeitsgänge werden in Maschinenzyklen ausgeführt. Die Art des Maschinenzyklus wird durch die Befehlssignale, die aus der Befehlsentschlüsselung hervorgehen oder von außen als Signale des Steuerbus an den Prozessor gelangen, gebildet. In jedem Maschinenzyklus entstehen durch Hinzufügen von Zeitsignalen zu den Befehlssignalen interne Steuersignale, die die Abarbeitung in Form von Registertransporten steuern.

| STS | = | BSI | ZSI |
|---|---|---|---|
| Steuersignal | | Befehlssignal | Zeitsignal |

Gleichzeitig werden zur Steuerung des Datentransfers mit den angeschlossenen Bausteinen äußere Steuersignale gebildet ($\overline{\text{RD}}$, $\overline{\text{WR}}$, $\overline{\text{IORQ}}$, $\overline{\text{MREQ}}$, $\overline{\text{M}_1}$, $\overline{\text{HALT}}$, $\overline{\text{BUSAK}}$). Die Abarbeitung eines Befehls setzt sich aus mehreren Maschinenzyklen zusammen. Die Folge dieser Maschinenzyklen ist eine Kombination der in Abschnitt 6.3. genannten Arten von Maschinenzyklen.

## 3.5. Befehlsliste des Prozessors *U 880*

### 3.5.1. Verwendete Abkürzungen bei der Befehlsbeschreibung

r – 8-Bit-Register des Registersatzes, A, B, C, D, E, H, L;
s – 8-Bit-Quellregister oder ein Speicherplatz oder eine 8-Bit-Zahl n
d – 8-Bit-Bestimmungsregister oder Speicherplatz

| BIT 7 | BIT 6 | BIT 5 | BIT 4 | BIT 3 | BIT 2 | BIT 1 | BIT 0 |
|---|---|---|---|---|---|---|---|

Bild 3.11 Bit-Numerierung innerhalb eines Bytes

n – 8-Bit-Zahl
nn – 16-Bit-Zahl
dd – 16-Bit-Bestimmungsregister
ss – 16-Bit-Quellregister
$s_b$ – Bit in einem speziellen 8-Bit-Register, b ist die Bit-Nr. (Bild 3.12);
Index L – der niederwertige Teil eines 16-Bit-Registers;
Index H – der höherwertige Teil eines 16-Bit-Registers.

– Steht ein Registername allein, z. B. A, so heißt das:
Inhalt von Register A.
– Steht ⟨HL⟩; so bedeutet das:
Inhalt der Speicherzelle, deren Adresse in HL steht.
– Steht ⟨nn⟩, so heißt das:
Inhalt der Speicherzelle, deren Adresse nn ist.

Bedeutung der Symbole für die Flagstellung (Bedeutung des Merkbits):
↕ Das Flag wird in Abhängigkeit vom Ergebnis der Operation beeinflußt.
· Das Flag bleibt unbeeinflußt.
0 Das Flag wird durch die Operation rückgesetzt.
1 Das Flag wird durch die Operation gesetzt.
v Das Flag wird in Abhängigkeit vom Überlauf des Ergebnisses beeinflußt.
P Das Flag wird in Abhängigkeit von der Parität des Ergebnisses beeinflußt.
X Das Flag ist beliebig.

## 3.5.2. Beschreibung der Befehle des Prozessors *U 880*

### 3.5.2.1. Transportbefehle

*Einzelworttransfer*

| | | C | S | Z | P/V | H | N |
|---|---|---|---|---|---|---|---|
| LD r, s | s → r | · | · | · | · | · | · |

Der Inhalt des Registers s oder eine Zahl n wird in ein Register r gebracht.

r kann sein: Register A, B, C, D, E, H, L.

s kann sein:
- Direktoperand n (8-Bit-Zahl);
- ⟨HL⟩, d. h. eine Speicherzelle, deren Adresse in HL steht;
- ⟨IX + d⟩, ⟨IY + d⟩, d. h. eine Speicherzelle, deren Adresse durch Indexrechnung ermittelt wird;
- Register A, B, C, D, E, H, L.

*Beispiel*

LD C, ⟨IX + 19H⟩

Der Inhalt von IX sei 25AFH. Durch den obigen Befehl wird der Inhalt der Zelle 25AFH + 19H = 25C8H nach Register C gebracht.

| LD d, s | s → d | C | S | Z | P/V | H | N |
|---|---|---|---|---|---|---|---|
| | | . | . | . | . | . | . |

Der Inhalt des Registers s oder eine Zahl n wird nach Zelle d gebracht.

s kann sein:
- Register A, B, C, D, E, H, L oder eine Zahl n;

d kann sein:
- ⟨HL⟩, d. h. eine Speicherzelle, deren Adresse in HL steht;
- ⟨IX + d⟩, ⟨IY + d⟩, d. h. eine Speicherzelle, deren Adresse durch Indexrechnung ermittelt wird;
- Register A, B, C, D, E, H, L.

*Beispiel*

LD ⟨HL⟩, 28H

Der Inhalt von HL sei 25A5H. Durch obigen Befehl wird die Zahl 28H nach Zelle 25A5H gebracht.

| LD A,s | s → A | C | S | Z | P/V | H | N | |
|---|---|---|---|---|---|---|---|---|
| | | . | ↕ | ↕ | IFF | 0 | 0 | bei s = I,R |
| | | . | . | . | . | . | 0 | bei s nicht I,R |

Der Inhalt einer Zelle s wird nach Register A gebracht. s kann sein:
- ⟨BC⟩, ⟨DE⟩, d.h. eine Speicherzelle, deren Adresse in BC oder DE steht;
- (nn), d.h. eine Speicherzelle, deren Adresse nn ist;
- Register I oder R.

*Beispiel*

LD A, ⟨DE⟩

Der Inhalt von DE sei 8A25H. Durch obigen Befehl wird der Inhalt von Zelle 8A25H nach Register A gebracht.

LD d, A   A → d   C S Z P/V H N
. . . . . .

Der Inhalt des Registers A wird nach Zelle d gebracht.

d kann sein:
- ⟨BC⟩, ⟨DE⟩, d.h. eine Speicherzelle, deren Adresse in BC oder DE steht;
- (nn), d.h. eine Speicherzelle, deren Adresse nn ist;
- Register I oder R.

*Beispiel*

LD ⟨25 H⟩, A

Der Inhalt des Registers A wird nach Zelle 25 H gebracht.

*Doppelworttransfer*

LD dd, nn   nn → dd   C S Z P/V H N
. . . . . .

Der Direktoperand nn wird in das Doppelregister dd gebracht.
dd kann sein: BC, DE, HL, SP, IX, IY.

*Beispiel*

LD HL, 28DEH

Die Zahl 28DEH wird nach Register HL gebracht.

LD dd, ⟨nn⟩   ⟨nn⟩ → dd   C S Z P/V H N
. . . . . .

Der Inhalt von Zelle nn und nn + 1 wird in das Doppelregister dd gebracht.
dd kann sein: BC, DE, HL, SP, IX, IY.

*Beispiel*

LD IX, ⟨8AH⟩

Der Inhalt von Zelle 8AH und 8 BH wird in das Indexregister IX gebracht.

LD ⟨nn⟩, ss   ss → ⟨nn⟩   C S Z P/V H N
. . . . . .

Der Inhalt des Doppelregisters ss wird in die Speicherzelle nn und nn + 1 gebracht.
ss kann sein: BC, DE, HL, SP, IX, IY.

*Beispiel*

LD ⟨20H⟩, SP

Der Inhalt des SP wird nach Zelle 20H gebracht.

LS SP, ss    ss → SP    C S Z P/V H N

. . . . . .

Der Inhalt des Doppelregisters ss wird in den Stackpointer SP gebracht.
ss kann sein: HL, IX, IY.

*Beispiel*

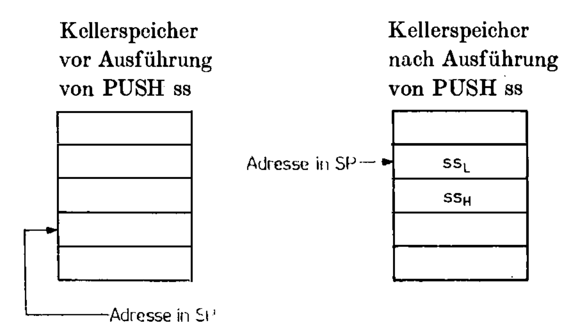
LD SP, IX

Der Inhalt des Indexregisters IX wird in den Stackpointer SP gebracht.

PUSH ss    $ss_H, ss_L$ → ⟨SP-1⟩, ⟨SP-2⟩    C S Z P/V H N

SP-2 → SP    . . . . . .

Der Inhalt des Doppelregisters ss wird in den Kellerspeicher gebracht. Der höherwertige Teil $ss_H$ kommt in die Zelle SP-1. Der niederwertige Teil in die Zelle SP-2. Nach Ausführung des Befehls ist der Inhalt des Stackpointers SP um 2 erniedrigt.




ss kann sein: BC, DE, HL, AF, IX, IY.

*Beispiel*

PUSH HL

Der Inhalt von SP ist 201H

Kellerspeicher vor PUSH HL

| | Adresse | Inhalt |
|---|---|---|
| | 1FD | |
| | 1FE | |
| | 1FF | |
| | 200 | |
| SP→ | 201 | |

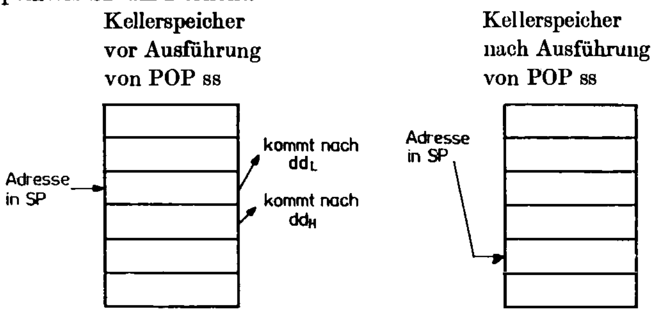
Kellerspeicher nach PUSH HL

| | Adresse | Inhalt |
|---|---|---|
| | 1FD | |
| | 1FE | |
| SP→ | 1FF | Inhalt von L |
| | 200 | Inhalt von H |
| | 201 | |

POP ss $\langle SP+1\rangle, \langle SP\rangle \rightarrow dd_H, dd_L$
$SP+2 \rightarrow SP$

| C | S | Z | P/V | H | N |
|---|---|---|---|---|---|
| . | . | . | . | . | . |

Aus dem Kellerspeicher werden 2 Byte in das Doppelregister dd gebracht. Das Byte aus Zelle SP kommt in den niederwertigen Teil von dd, das Byte aus Zelle SP+1 in den höherwertigen Teil von dd. Nach Ausführung des Befehls ist der Inhalt des Stackpointers SP um 2 erhöht.

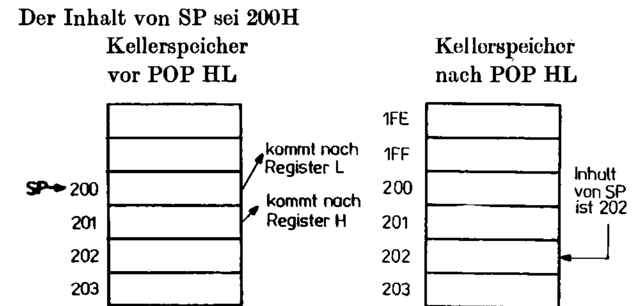



ss kann sein: BC, DE, HL, AF, IX, IY.

*Beispiel*

POP HL

Der Inhalt von SP sei 200H




*Doppelworttransfer-Umtauschbefehle*

EX DE, HL  DE ↔ HL

| C | S | Z | P/V | H | N |
|---|---|---|---|---|---|
| . | . | . | . | . | . |

Der Inhalt des Registerpaares DE wird mit dem Inhalt des Registerpaares HL vertauscht.

EX, AF, AF′  AF ↔ A′F′

| C | S | Z | P/V | H | N |
|---|---|---|---|---|---|
| . | . | . | . | . | . |

Die Inhalte der Register A und F werden mit den Inhalten der Register A′ und F′ vertauscht, und zwar A mit A′ und F mit F′.

| | | C S Z P/V H N |
|---|---|---|
| EXX | BC ↔ B′C′ | . . . . . . |
| | DE ↔ D′E′ | |
| | HL ↔ H′L′ | |

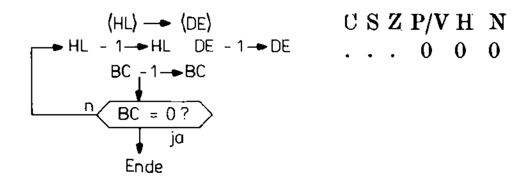
Es werden die Inhalte der Register B mit B′, C mit C′, D mit D′, E mit E′, H mit H′ und L mit L′ vertauscht.

| | C S Z P/V H N |
|---|---|
| EX ⟨SP⟩, ss $ss_H, ss_L$ ↔ ⟨SP+1⟩, ⟨SP⟩ | . . . . . . |

Der Inhalt des Doppelregisters ss wird mit dem Inhalt von 2 Zellen des Kellerspeichers vertauscht. Es wird dabei der niederwertige Teil $ss_L$ des Doppelregisters mit dem Inhalt der Zelle, deren Adresse in SP steht, und der höherwertige Teil $ss_H$ mit der nächsten Zelle (Adresse SP+1) vertauscht. Am Ende steht in SP der gleiche Wert wie vorher.
ss kann sein: HL, IX, IY.

*Blocktransfer*

LDIR

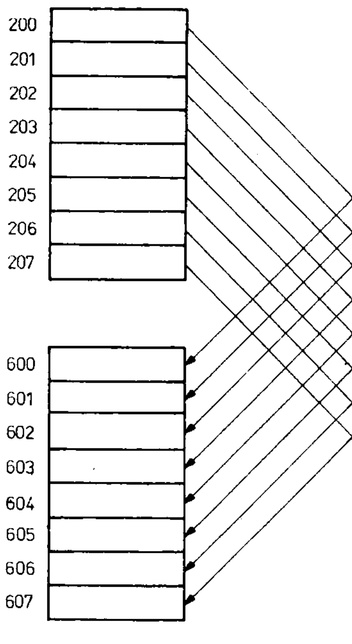



| C S Z P/V H N |
|---|
| . . . 0 0 0 |

Es wird der Inhalt eines Speicherbereiches, dessen Anfangsadresse in HL und dessen Blocklänge (Anzahl der Zellen des Speicherbereiches) in BC steht, in einem Speicherbereich mit der Anfangsadresse, die in DE steht, gespeichert.

*Beispiel*

Der Inhalt von HL sei 200, der von DE 600 und der von BC 8. Durch LDIR wird der Inhalt der Zellen 200 bis 207 in den Zellen 600 bis 607 gespeichert.




| LDI | ⟨HL⟩ → ⟨DE⟩ | C S Z P/V H N |
|---|---|---|
| | HL + 1 → HL DE + 1 → DE | . . . ↕ 0 0 |
| | BC − 1 → BC | |

Dieser Befehl dient zur Umspeicherung eines Speicherbereiches, dessen Anfangsadresse in HL und dessen Blocklänge in BC steht, in einen Speicherbereich, dessen Anfangsadresse in DE steht. Durch eine einmalige Abarbeitung des Befehls wird der Inhalt der Zelle, deren Adresse in HL steht, in die Speicherzelle gebracht, deren Adresse in DE steht. Anschließend werden die Adressen in HL und DE um 1 erhöht und der Inhalt von BC um 1 erniedrigt. Ist BC − 1 = 0, so wird das Flag P/V = 0, sonst wird P/V = 1 gesetzt. Durch mehrmaliges Anwenden dieses Befehls läßt sich der Inhalt eines Speicherbereiches in einen anderen Speicherbereich umspeichern. Dabei kann entweder bei BC = 0 oder bei einer anderen Bedingung abgebrochen werden.

*Beispiel*
Der Inhalt von HL sei 300, der von DE 500 und der von BC sei 12.

Bei der 1. Befehlsabarbeitung von LDI kommt der Inhalt von Zelle 300 nach Zelle 500. Bei der 2. Abarbeitung von LDI wird der Inhalt von Zelle 301 nach Zelle 501 gebracht usw. Nach 12 Durchläufen (Inhalt von BC) ist der Inhalt von BC = 0, was als Endbedingung genommen werden kann.

LDDR

(HL) → (DE)
→ HL - 1 → HL   DE - 1 → DE
BC - 1 → BC
n ⟨ BC = 0? ⟩
ja
Ende

| C | S | Z | P/V | H | N |
|---|---|---|---|---|---|
| . | . | . | 0 | 0 | 0 |

Es wird der Inhalt eines Speicherbereiches, dessen Endadresse in HL und dessen Blocklänge in BC steht, in einen Speicherbereich, dessen Endadresse in DE steht, gebracht.

*Beispiel*

Der Inhalt von HL sei 200H, der von DE 600H und der von BC 8H. Durch LDDR gelangt der Inhalt der Zellen 1F9-200H in die Zellen 5F9-600H.

LDD ⟨HL⟩ → ⟨DE⟩
HL - 1 → HL DE - 1 → DE
BC - 1 → BC

| C | S | Z | P/V | H | N |
|---|---|---|---|---|---|
| . | . | . | ↕ | 0 | 0 |

Dieser Befehl dient zur Umspeicherung eines Speicherbereiches, dessen Endadresse in HL und dessen Blocklänge in BC steht, in einen Speicherbereich, dessen Endadresse in DE steht.

Durch eine einmalige Abarbeitung des Befehls wird der Inhalt der Zelle, deren Adresse in HL steht, in die Speicherzelle gebracht, deren Adresse in DE steht. Anschließend werden die Adressen in HL und DE um 1 erniedrigt und der Inhalt von BC um 1 erniedrigt. Ist BC - 1 = 0, so wird das Flag P/V = 0, sonst wird P/V = 1 gesetzt. Durch mehrmaliges Anwenden dieses Befehls läßt sich der Inhalt eines Speicherbereiches in einen anderen Speicherbereich umspeichern. Dabei kann entweder bei BC = 0 oder bei einer anderen Bedingung abgebrochen werden.

### 3.5.2.2. Rechen- und logische Operationen mit einem Operand

*Akkumulator- und C-Bit-Befehle*

CPL $\overline{A} \to A$

| C | S | Z | P/V | H | N |
|---|---|---|---|---|---|
| . | . | . | . | 1 | 1 |

Der Inhalt des Registers A wird bitweise negiert.

*Beispiel*
Der Inhalt von Register A sei 11001110.
Nach Ausführung des Befehls CPL ist der Inhalt des Registers A 00110001.

NEG $\overline{A} + 1 \to A$ oder $0 - A \to A$

| C | S | Z | P/V | H | N |
|---|---|---|---|---|---|
| ↕ | ↕ | ↕ | V | ↕ | 1 |

Vom Inhalt des Registers A wird das Zweierkomplement gebildet.

*Beispiel*
Der Inhalt von Register A sei 11001110.
Nach Ausführung des Befehls NEG ist der Inhalt des Registers A 00110010.

CCF $\overline{C} \to C$

| C | S | Z | P/V | H | N |
|---|---|---|---|---|---|
| ↕ | . | . | . | ↕ | 0 |

Der Inhalt des C-Bits wird negiert.
Im H-Bit wird der vorherige Wert des C-Bit gespeichert.

SCF $1 \to C$

| C | S | Z | P/V | H | N |
|---|---|---|---|---|---|
| 1 | . | . | . | 0 | 0 |

Der Inhalt des C-Bits wird 1 gesetzt.

DAA

| C | S | Z | P/V | H | N |
|---|---|---|---|---|---|
| ↕ | ↕ | ↕ | P | ↕ | . |

Der Befehl DAA dient im Zusammenhang mit der Addition und der Subtraktion von Dualzahlen zur Berechnung von Summen oder Differenzen zweier im BCD-Code dargestellten Zahlen.

*Beispiel*
Im Register A stehe die Zahl 28 im BCD-Code, d. h., der Inhalt von A ist 00101000. Im Register B stehe die Zahl 17 im BCD-Code, d. h., B = 00010111. Nach der dualen Addition der Inhalte von Register A und B steht im Register A 00111111.

Das ist jedoch nicht die BCD-Code-Darstellung von 28 + 17 = 45. Der Befehl DAA verändert A = 00111111 in den Wert A = 01000101 (= 45 in BCD-Darstellung).

*Einzelwortbefehle*

INC d  d + 1 → d

| C | S | Z | P/V | H | N |
|---|---|---|---|---|---|
| | ↕ | ↕ | V | ↕ | 0 |

Der Inhalt der Zelle oder des Registers d wird um 1 erhöht.
d kann sein:
- Register A, B, C, D, E, H, L;
- ⟨HL⟩, d. h. eine Speicherzelle, deren Adresse in HL steht;
- ⟨IX + d⟩, ⟨IY + d⟩, d. h. Inhalt einer Speicherzelle, deren Adresse durch Indexrechnung ermittelt wird.

*Beispiel*

Der Inhalt vom Indexregister IY sei 1A5H.
Durch den Befehl INC ⟨IY + 17H⟩ wird der Inhalt von Zelle 1A5H + 17H = 1BCH um 1 erhöht.

DEC d  d − 1 → d

| C | S | Z | P/V | H | N |
|---|---|---|---|---|---|
| . | ↕ | ↕ | V | ↕ | 1 |

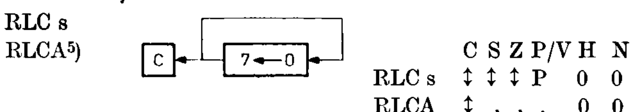
Der Inhalt der Zelle oder des Registers d wird um 1 erniedrigt,
d kann sein:
- Register A, B, C, D, E, H, L;
- ⟨HL⟩, d. h. eine Speicherzelle, deren Adresse in HL steht;
- ⟨IX + d⟩, ⟨IY + d⟩, d. h. eine Speicherzelle, deren Adresse durch Indexrechnung ermittelt wird.

*Beispiel*

Der Inhalt des Doppelregisters HL sei 800.
Durch den Befehl DEC ⟨HL⟩ wird der Inhalt der Zelle 800 um 1 erniedrigt.

*Verschiebebefehle*

RLC s
RLCA[5])




| | C | S | Z | P/V | H | N |
|---|---|---|---|---|---|---|
| RLC s | ↕ | ↕ | ↕ | P | 0 | 0 |
| RLCA | ↕ | . | . | . | 0 | 0 |

[5]) RLCA führt dieselbe Funktion wie RLC A aus, ist jedoch ein 1-Byte-Befehl.

Der Inhalt des Registers oder der Zelle s wird um eine Stelle nach links verschoben. Das aus Bit 7 (Zählung von rechts nach links) heraustretende Bit wird in das C-Bit und in Bit 0 eingetragen.

s kann sein:
- Register A, B, C, D, E, H, L;
- ⟨HL⟩, d. h. eine Speicherzelle, deren Adresse in HL steht;
- ⟨IX + d⟩, ⟨IY + d⟩, d. h. eine Speicherzelle, deren Adresse durch Indexrechnung ermittelt wird.

RL s
RLA[6])

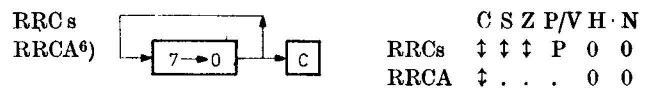
C ← 7←0 ←

| | C | S | Z | P/V | H | N |
|---|---|---|---|---|---|---|
| RLs | ↕ | ↕ | ↕ | P | 0 | 0 |
| RLA | ↕ | . | . | . | 0 | 0 |

Der Inhalt des Registers s oder der Zelle s wird zusammen mit dem C-Bit um eine Stelle nach links verschoben. Das aus Bit 7 kommende Bit wird in das C-Bit und das C-Bit in Bit 0 eingetragen.

s kann sein:
- Register A, B, C, D, E, H, L;
- ⟨HL⟩, d. h. eine Speicherzelle, deren Adresse in HL steht;
- ⟨IX + d⟩, ⟨IY + d⟩, d. h. eine Speicherzelle, deren Adresse durch Indexrechnung ermittelt wird.

RRC s
RRCA[6])



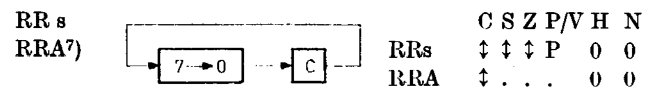

| | C | S | Z | P/V | H | N |
|---|---|---|---|---|---|---|
| RRCs | ↕ | ↕ | ↕ | P | 0 | 0 |
| RRCA | ↕ | . | . | . | 0 | 0 |

Der Inhalt des Registers s oder der Zelle s wird um eine Stelle nach rechts verschoben. Das aus Bit 0 heraustretende Bit wird in das C-Bit und in Bit 7 eingetragen.

s kann sein:
- Register A, B, C, D, E, H, L;
- ⟨HL⟩, d. h. eine Speicherzelle, deren Adresse in HL steht;
- ⟨IX + d⟩, ⟨IY + d⟩, d. h. eine Speicherzelle, deren Adresse durch Indexrechnung ermittelt wird.

[6]) RLA und RRCA führen dieselbe Verschiebung wie RL A und RRC A aus, sind aber 1-Byte-Befehle.

RR s
RRA[7])




| | C | S | Z | P/V | H | N |
|---|---|---|---|---|---|---|
| RRs | ↕ | ↕ | ↕ | P | 0 | 0 |
| RRA | ↕ | . | . | . | 0 | 0 |

Der Inhalt des Registers s oder der Zelle s wird zusammen mit dem C-Bit um eine Stelle nach rechts verschoben. Bit 0 kommt ins C-Bit und das C-Bit nach Bit 7.

s kann sein:
- Register A, B, C, D, E, H, L;
- ⟨HL⟩, d. h. eine Speicherzelle, deren Adresse in HL steht;
- ⟨IX + d⟩, ⟨IY + d⟩, d. h. eine Speicherzelle, deren Adresse durch Indexrechnung ermittelt wird.

SLA s

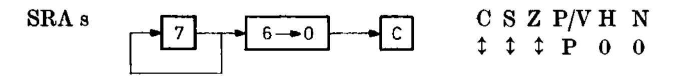



| C | S | Z | P/V | H | N |
|---|---|---|---|---|---|
| ↕ | ↕ | ↕ | P | 0 | 0 |

Dieser Befehl bewirkt eine arithmetische Linksverschiebung des Registers oder der Speicherzelle s. Das aus Bit 7 heraustretende Bit wird in das C-Bit eingetragen.
In Bit 0 wird eine 0 eingetragen. Der Befehl entspricht der Multiplikation des Registerinhalts mit 2.

s kann sein:
- Register A, B, C, D, E, H, L;
- ⟨HL⟩, d. h. eine Speicherzelle, deren Adresse in HL steht;
- ⟨IX + d⟩, ⟨IY + d⟩, d. h. eine Speicherzelle, deren Adresse durch Indexrechnung ermittelt wird.

SRA s




| C | S | Z | P/V | H | N |
|---|---|---|---|---|---|
| ↕ | ↕ | ↕ | P | 0 | 0 |

Dieser Befehl bewirkt eine arithmetische Rechtsverschiebung des Registers oder der Speicherzelle s. Bit 7 bleibt erhalten. In Bit 6 wird Bit 7 eingetragen, Bit 0 kommt ins C-Bit.

s kann sein:
- Register A, B, C, D, E, H, L;
- ⟨HL⟩, d.h. eine Speicherzelle, deren Adresse in HL steht;

7) RRA führt dieselbe Verschiebung wie RR A aus, ist aber ein 1-Byte-Befehl.

– ⟨IX + d⟩, ⟨IY + d⟩, d. h. eine Speicherzelle, deren Adresse durch Indexrechnung ermittelt wird.

SRL s  0 → [7 → 0] → [C]

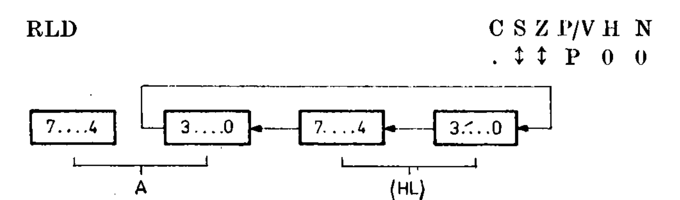
| C | S | Z | P/V | H | N |
|---|---|---|---|---|---|
| ↕ | ↕ | ↕ | P | 0 | 0 |

Dieser Befehl bewirkt eine Rechtsverschiebung des Registers oder der Zelle s. Dabei wird in Stelle 7 eine 0 und in das C-Bit Bit 0 eingetragen.

s kann sein:
- Register A, B, C, D, E, H, L;
- ⟨HL⟩, d. h. eine Speicherzelle, deren Adresse in HL steht;
- ⟨IX + d⟩, ⟨IY + d⟩ d. h. eine Speicherzelle, deren Adresse durch Indexrechnung ermittelt wird.

RLD

| C | S | Z | P/V | H | N |
|---|---|---|---|---|---|
| . | ↕ | ↕ | P | 0 | 0 |



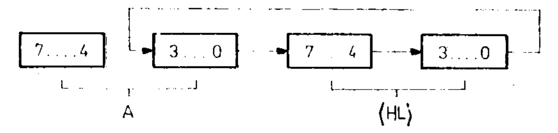

Dieser Befehl bewirkt eine Linksverschiebung um eine Tetrade (4 Bit) des Registers A und einer Speicherzelle, deren Adresse in HL steht.

Dabei werden
- die niederwertige Tetrade von A in die niederwertige Tetrade der Speicherzelle,
- die niederwertige Tetrade der Speicherzelle in die höherwertige Tetrade der Speicherzelle und
- die höherwertige Tetrade der Speicherzelle in die niederwertige Tetrade des Registers A

eingetragen.

RRD

| C | S | Z | P/V | H | N |
|---|---|---|---|---|---|
| . | ↕ | ↕ | P | 0 | 0 |




Dieser Befehl bewirkt eine Rechtsverschiebung um eine Tetrade (4 Bit) des Registers A und einer Speicherzelle, deren Adresse in HL steht.
Dabei wird
- die niederwertige Tetrade von A in die höherwertige Tetrade der Speicherzelle,
- die höherwertige Tetrade der Speicherzelle in die niederwertige Tetrade der Speicherzelle und die
- niederwertige Tetrade der Speicherzelle in die niederwertige Tetrade des Registers A

eingetragen.

*Beispiel (zu den Verschiebebefehlen)*
Im Register D stehe die Bit-Folge 01110011 und im C-Bit eine 1.

| C-Bit | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |

Register D

Befehl: Information im D-Register und C-Bit nach dem Befehl:

| Befehl | C-Bit | Register D |
|---|---|---|
| RLC D | 0 | 11100110 |
| RL D | 0 | 11100111 |
| RRC D | 1 | 10111001 |
| RR D | 1 | 10111001 |
| SLA D | 0 | 11100110 |
| SRA D | 1 | 00111001 |
| SRL D | 1 | 00111001 |

*Bit-Befehle*

Bit b, s $s_b \rightarrow Z$

| C | S | Z | P/V | H | N |
|---|---|---|---|---|---|
| . | X | ↕ | X | 1 | 0 |

Das Bit b des Registers oder der Speicherzelle s wird in negierter Form in das Z-Flag gebracht, b ist eine Zahl zwischen 0 und 7.

s kann sein: – Register A, B, C, D, E, H, L;
– ⟨HL⟩, d. h. eine Speicherzelle, deren Adresse in HL steht;
– ⟨IX + d⟩, ⟨IY + d⟩, d. h. eine Speicherzelle, deren Adresse durch Indexrechnung ermittelt wird.

Bit-Zählung

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|

*Beispiel*

Bit 3, ⟨HL⟩

Bit 3 der Speicherzelle, deren Adresse in HL steht, wird in negierter Form in das Z-Flag gebracht.

- In HL steht die Adresse 80H.
- In der Zelle 80H steht 11101110.
- Durch Bit 3, ⟨HL⟩ wird eine 0 in das Z-Flag gebracht.

SET b, s $1 \rightarrow s_b$

| C | S | Z | P/V | H | N |
|---|---|---|---|---|---|
| . | . | . | . | . | . |

Das Bit b des Registers oder der Speicherzelle s wird 1 gesetzt, b ist eine Zahl zwischen 0 und 7.

s kann sein: – Register A, B, C, D, E, H, L;
– ⟨HL⟩, d. h. eine Speicherzelle, deren Adresse in HL steht;
– ⟨IX + d⟩, ⟨IY + d⟩, d. h. eine Speicherzelle, deren Adresse durch Indexrechnung ermittelt wird.

RES b, s $0 \rightarrow s_b$

| C | S | Z | P/V | H | N |
|---|---|---|---|---|---|
| . | . | . | . | . | . |

Das Bit b des Registers oder der Speicherzelle s wird 0 gesetzt.

s kann sein: – Register A, B, C, D, E, H, L;
– ⟨HL⟩, d. h. eine Speicherzelle, deren Adresse in HL steht;
– ⟨IX + d⟩, ⟨IY + d⟩, d. h. eine Speicherzelle, deren Adresse durch Indexrechnung ermittelt wird.

*Doppelwortbefehle*

| INC dd | dd + 1 → dd | C | S | Z | P/V | H | N |
|---|---|---|---|---|---|---|---|
| | | · | · | · | · | · | · |

Der Inhalt des Doppelregisters dd wird um 1 erhöht.
dd kann sein: BC, DE, HL, SP, IX, IY.

| DEC dd | dd − 1 → dd | C | S | Z | P/V | H | N |
|---|---|---|---|---|---|---|---|
| | | · | · | · | · | · | · |

Der Inhalt des Doppelregisters dd wird um 1 erniedrigt. dd kann sein: BC, DE, HL, SP, IX, IY.

### 3.5.2.3. Rechen- und logische Operationen mit zwei Operanden

*1-Wort-Befehle*

| ADD s | A + s → A | C | S | Z | P/V | H | N |
|---|---|---|---|---|---|---|---|
| | | ↕ | ↕ | ↕ | V | ↕ | 0 |

Der Inhalt des Akkumulators und der Inhalt des Registers oder der Zelle s werden addiert, und das Ergebnis wird ins Register A gebracht.

s kann sein:
- Direktoperand n (8-Bit-Zahl);
- ⟨HL⟩, d. h. eine Speicherzelle, deren Adresse in HL steht;
- ⟨IX + d⟩, ⟨IY + d⟩, d. h. eine durch Indexierung ermittelte Speicherzelle;
- Register A, B, C, D, E, H, L.

| ADC s | A + s + C → A | C | S | Z | P/V | H | N |
|---|---|---|---|---|---|---|---|
| | | ↕ | ↕ | ↕ | V | 0 | ↕ |

Der Inhalt des Registers A und der Inhalt des Registers oder der Zelle s werden addiert. Zu diesem Ergebnis wird das C-Bit in die niedrigste Stelle addiert. Das Gesamtergebnis kommt in das Register A. s kann sein: siehe 1-Wort-Befehl ADD s.

| SUB s | A − s → A | C | S | Z | P/V | H | N |
|---|---|---|---|---|---|---|---|
| | | ↕ | ↕ | ↕ | V | ↕ | 1 |

Der Inhalt der Zelle oder des Registers s wird vom Register A subtrahiert. Das Ergebnis kommt in das Register A.
s kann sein: siehe 1-Wort-Befehl ADD s.

SBC s $A - s - C \rightarrow A$

| C | S | Z | P/V | H | N |
|---|---|---|---|---|---|
| $\updownarrow$ | $\updownarrow$ | $\updownarrow$ | V | $\updownarrow$ | 1 |

Der Inhalt des Registers oder der Zelle s wird vom Inhalt des A-Registers subtrahiert. Von der niedrigsten Stelle des Ergebnisses wird das C-Bit subtrahiert. Das letzte Ergebnis kommt in das Register A.
s kann sein: siehe 1-Wort-Befehl ADD s.

AND s $A \wedge s \rightarrow A$

| C | S | Z | P/V | H | N |
|---|---|---|---|---|---|
| 0 | $\updownarrow$ | $\updownarrow$ | P | 1 | 1 |

Der Inhalt des Registers A und der Inhalt des Registers oder der Zelle s werden bitweise durch ein logisches UND verknüpft. Das Ergebnis kommt in das Register A.

s kann sein: siehe 1-Wort-Befehl ADD s.

OR s $A \vee s \rightarrow A$

| C | S | Z | P/V | H | N |
|---|---|---|---|---|---|
| 0 | $\updownarrow$ | $\updownarrow$ | P | 1 | 1 |

Der Inhalt des Registers A und der Inhalt des Registers oder der Zelle s werden bitweise durch ein logisches ODER verknüpft. Das Ergebnis kommt in das Register A.

s kann sein: siehe 1-Wort-Befehl ADD s.

XOR s $A \oplus s \rightarrow A$

| C | S | Z | P/V | H | N |
|---|---|---|---|---|---|
| 0 | $\updownarrow$ | $\updownarrow$ | P | 1 | 0 |

Der Inhalt des Registers A und der Inhalt des Registers oder der Zelle s werden bitweise durch ein logisches EXKLUSIV-ODER verknüpft. Das Ergebnis kommt in das Register A. s kann sein: siehe 1-Wort-Befehl ADD s.

CP s

| C | S | Z | P/V | H | N |
|---|---|---|---|---|---|
| $\updownarrow$ | $\updownarrow$ | $\updownarrow$ | V | $\updownarrow$ | 1 |

Der Inhalt des Registers A wird mit dem Inhalt des Registers oder der Zelle s verglichen. Der Vergleich erfolgt durch die Bildung der Differenz A — s. Bei Gleichheit wird das Z-Bit gesetzt, bei Ungleichheit rückgesetzt.

s kann sein: siehe 1-Wort-Befehl ADD s.

*Doppelwortbefehle*

ADD HL, ss  HL + ss → HL

| C | S | Z | P/V | H | N |
|---|---|---|---|---|---|
| ↕ | . | . | . | ↕ | 0 |

Der Inhalt des Doppelregisters HL wird mit dem Inhalt des Doppelregisters ss addiert. Das Ergebnis kommt nach HL. Bit H wird mit dem Übertrag aus Bit 11 gesetzt.
ss kann sein: BC, DE, HL, SP.

ADC HL, ss  HL + ss + C → HL

| C | S | Z | P/V | H | N |
|---|---|---|---|---|---|
| ↕ | ↕ | ↕ | V | ↕ | 0 |

Der Inhalt des Doppelregisters HL wird mit dem Inhalt des Doppelregisters ss addiert. Zum Ergebnis wird das C-Bit in die niedrigste Stelle addiert. Das letzte Ergebnis kommt nach HL. Bit H wird mit dem Übertrag aus Bit 11 gesetzt. ss kann sein: BC, DE, HL, SP.

SBC HL, ss  HL − ss − C → HL

| C | S | Z | P/V | H | N |
|---|---|---|---|---|---|
| ↕ | ↕ | ↕ | V | ↕ | 1 |

Der Inhalt des Doppelregisters ss wird vom Doppelregister HL subtrahiert. Anschließend wird davon das C-Bit in der letzten Stelle subtrahiert. Das Ergebnis kommt nach HL, Bit H wird vom Übertrag von Bit 12 gesetzt.
ss kann sein: BC, DE, HL, SP.

ADD IX, ss  IX + ss → IX

| C | S | Z | P/V | H | N |
|---|---|---|---|---|---|
| ↕ | . | . | . | ↕ | 0 |

Der Inhalt des Doppelregisters ss wird zum Inhalt des Indexregisters IX addiert. Das Ergebnis kommt nach IX. Bit H wird mit dem Übertrag aus Bit 11 gesetzt.
ss kann sein: BC, DE, IX, SP.

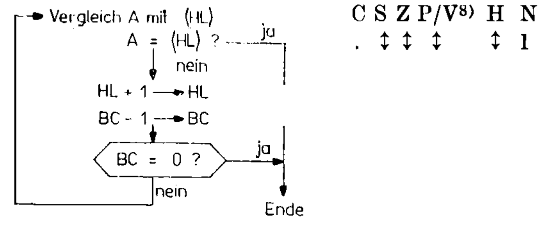
ADD IY, ss  IY + ss → IY

| C | S | Z | P/V | H | N |
|---|---|---|---|---|---|
| ↕ | . | . | . | ↕ | 0 |

Der Inhalt des Doppelregisters ss wird zum Inhalt des Indexregisters IY addiert. Das Ergebnis kommt nach IY. Bit H wird mit dem Übertrag aus Bit 11 gesetzt.
ss kann sein: BC, DE, IY, SP.

### 3.5.2.4. Rechen- und logische Operationen mit mehreren Operanden

CPIR




| C | S | Z | P/V[8] | H | N |
|---|---|---|---|---|---|
| . | ↕ | ↕ | ↕ | ↕ | 1 |

Der Inhalt eines Speicherbereiches, dessen Anfangsadresse in HL und dessen Länge in BC steht, wird nach einer Information im Register A durchsucht. Befindet sich die Information in einer Zelle des Speicherbereiches, so ist der Befehl an dieser Stelle beendet. Befindet sich die Information nicht in dem gesuchten Speicherbereich, so wird der Befehl nach der letzten Zelle des Bereiches beendet. Wird die Information im Speicherbereich gefunden, so wird trotzdem HL um 1 erhöht und BC um 1 erniedrigt.

Im Flagregister steht das Ergebnis der letzten Vergleichsoperation. Das Bit P/V ist 1, wenn $BC - 1 \neq 0$, sonst 0.

*Beispiel*

Die Zellen 200 H bis 205 H sollen nach der Bit-Folge 00000111 durchsucht werden.

In den Zellen 200 H bis 205 H stehen:

200H 11111111
201H 00000000
202H 01000000
203H 00000111
204H 00000000
205H 10000001

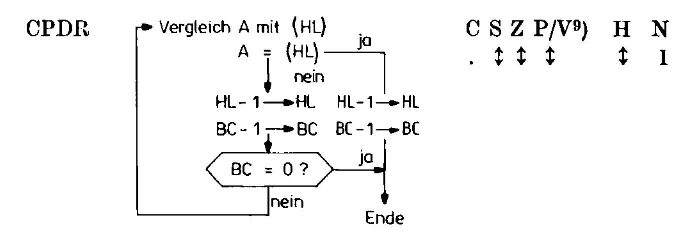
Dazu bringt man die Bit-Folge 00000111 in das Register A, die Adresse 200H ins Registerpaar HL und die Anzahl 6 ins Registerpaar BC. Nach der Ausführung des Befehls CPIR steht in HL 204H, und das Z-Bit ist gesetzt.

8) Wenn BC zum Befehlsende 0 ist, dann wird P/V = 0; ist $BC \neq 0$, so wird P/V = 1 gesetzt.

Würde in Zelle 203 die Bit-Folge 01010101 stehen, so stünde nach Abarbeitung des Befehls CPIR in HL 206H, und das Z-Bit würde rückgesetzt sein.

| CPI | Vergleich A mit ⟨HL⟩ | C | S | Z | P/V[9]) | H | N |
|---|---|---|---|---|---|---|---|
| | HL + 1 → HL | . | $\updownarrow$ | $\updownarrow$ | $\updownarrow$ | $\updownarrow$ | 1 |
| | BC − 1 → BC | | | | | | |

Der Inhalt des Speicherplatzes, dessen Adresse in HL steht, wird mit dem Inhalt des A-Registers verglichen und das Flagregister gesetzt. Das P/V-Flag ist 1, wenn BC − 1 ≠ 0, sonst 0. Anschließend wird der Inhalt von HL um 1 erhöht und der Inhalt von BC um 1 erniedrigt. Mit Hilfe des Befehls CPI läßt sich ein Speicherbereich auf eine Bit-Folge, die im A-Register steht, durchsuchen, wobei die Abbruchbedingung frei wählbar ist.




Der Inhalt eines Speicherbereiches, dessen Endadresse in HL und dessen Länge in BC steht, wird nach einer Information, die im Register A steht, durchsucht. Befindet sich die Information in einer Zelle des Speicherbereiches, so wird der Befehl an dieser Stelle beendet. Steht die Information nicht in dem gesuchten Speicherbereich, so endet der Befehl nach der ersten Zelle des Bereiches. Wird die Information im Speicherbereich gefunden, so werden trotzdem HL und BC um 1 erniedrigt.
Im Flagregister steht das Ergebnis der letzten Vergleichsoperation. Das Bit P/V ist 1, wenn BC − 1 ≠ 0, sonst 0.

*Beispiel*
Die Zellen 200H bis 205H sollen nach der Bit-Folge 00000111 durchsucht werden.

9) Wenn BC nach Befehlsausführung 0 ist, dann wird P/V = 0; ist BC ≠ 0, so wird P/V = 1 gesetzt.

In den Zellen 200H bis 205H stehen:

200H 11111111
201H 00000000
202H 01000000
203H 00000111
204H 00000000
205H 10000001

Dazu bringt man die obige Bit-Folge in das Register A, die Adresse 205H in das Register HL und die Anzahl 6 in das Registerpaar BC. Nach Ausführung des Befehls CPDR steht in HL 202H, und das Z-Bit ist gesetzt.
Würde sich in Zelle 203H die Bit-Folge 01010101 befinden, so stünde nach Abarbeitung des Befehls CPDR in HL 1FFH, und das Z-Bit würde rückgesetzt sein.

| CPD | Vergleich A mit ⟨HL⟩ | C S Z P/V[10] H N |
|---|---|---|
| | HL − 1 → HL | . ↕ ↕ ↕ ↕ 1 |
| | BC − 1 → BC | |

Der Inhalt des Speicherplatzes, dessen Adresse in HL steht, wird mit dem Inhalt des A-Registers verglichen und das Flagregister gesetzt. Das P/V-Flag ist 1, wenn $BC - 1 \neq 0$, sonst 0. Anschließend werden der Inhalt von, HL und der von BC um 1 erniedrigt. Mit Hilfe des Befehls CPD läßt sich ein Speicherbereich auf eine Bit-Folge, die im A-Register steht, durchsuchen, wobei die Abbruchbedingung frei wählbar ist.

### 3.5.2.5. Sprungbefehle

| JP nn | nn → PC | C S Z P/V H N |
|---|---|---|
| | | . . . . . . |

Die Adresse nn wird in den Befehlszähler PC gebracht. Der nächste abzuarbeitende Befehl ist damit der Befehl aus Zelle nn. Man sagt, der Rechner führt einen Sprung in die Zelle nn aus.

| JP cc, nn | Wenn cc = true, nn → PC | C S Z P/V H N |
|---|---|---|
| | | . . . . . . |

Wenn die Bedingung cc erfüllt ist, dann wird die Adresse nn in den Befehlszähler PC gebracht.

10) s. Fußnote 9, S. 64

cc kann sein: – NZ Z-Bit rückgesetzt,
– Z Z-Bit gesetzt,
– NC C-Bit rückgesetzt,
– C C-Bit gesetzt,
– PO Paritätsbit auf ungerade gesetzt,
– PE Paritätsbit auf gerade gesetzt,
– P Vorzeichenbit auf positiv gesetzt,
– M Vorzeichenbit auf negativ gesetzt.

| JR e | PC + e → PC | C | S | Z | P/V | H | N |
|---|---|---|---|---|---|---|---|
| | | . | . | . | . | . | . |

Der Befehlszähler PC wird um die Zahl e verändert. e ist eine 8-Bit-Zahl im Zweierkomplement, d. h., e kann auch negativ sein. Ausgangspunkt für PC + e → PC ist die Adresse des Operationscodes des Befehls. Der Rechner führt einen Sprung um e Zellen aus. Die Zahl e steht als e — 2 in der Zelle nach dem Operationscode.

*Beispiel*

Von Zelle 100 soll ein Sprung nach 105 erfolgen.
Befehl: JR 5

```
┌─100   18    ← Operationscode des Befehls JR
│ 101    3    ← e − 2
│ 102   ——
│ 103   ———
│ 104   ———
└→105   ———   ← PC-Stand nach dem Befehl.
```

| JR kk, e | Wenn kk = true, PC + e → PC | C | S | Z | P/V | H | N |
|---|---|---|---|---|---|---|---|
| | wenn kk = not true, Leerbefehl | . | . | . | . | . | . |

Wenn die Bedingung kk erfüllt ist, dann führt der Prozessor einen Sprung um e Zellen aus.

kk kann sein: – NZ Z-Bit rückgesetzt,
– Z Z-Bit gesetzt,
– NC C-Bit rückgesetzt,
– C C-Bit gesetzt.

| JP (ss) | ss → PC | C | S | Z | P/V | H | N |
|---|---|---|---|---|---|---|---|
| | | . | . | . | . | . | . |

Der Inhalt des Doppelregisters ss wird nach PC gebracht. Der

Rechner führt einen Sprung in die Zelle aus, deren Adresse in ss steht.
ss kann sein: HL, IX, IY.

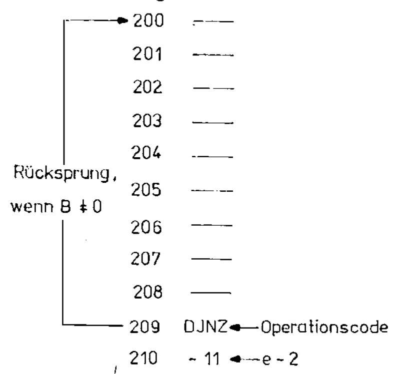
DJNZ e

B - 1 → B
B = 0 ? — nein → PC + e → PC
ja ↓ nächster Befehl

| C | S | Z | P/V | H | N |
|---|---|---|---|---|---|
| . | . | . | . | . | . |

Solange $B - 1 \neq 0$ ist, führt der Rechner einen Sprung um e Zellen aus. e ist eine 8-Bit-Zahl im Zweierkomplement, d. h., e kann auch negativ sein. Ausgangspunkt für $PC + e \rightarrow PC$ ist die Adresse des Operationscodes des Befehls. Die Zahl e steht als $e - 2$ in der Zelle nach dem Operationscode.

*Beispiel*
Eine Programmschleife von Zelle 200 bis 209 soll 10mal durchlaufen werden. Dazu bringt man eine 10 in das Register B. Der Befehl, der diese 10fache Programmschleife realisiert, lautet DJNZ − 9.
Aufbau der Programmschleife:




### 3.5.2.6. Unterprogrammbefehle

*Unterprogrammrufe*

| | | | C | S | Z | P/V | H |
|---|---|---|---|---|---|---|---|
| CALL nn | $PC_H$, $PC_L$ | → ⟨SP − 1⟩, ⟨SP − 2⟩ | | | | | |
| | SP−2 | → SP | . | . | . | . | . |
| | nn | → PC | | | | | |

Der CALL-Befehl realisiert einen Sprung in ein Unterprogramm, dessen Startadresse nn ist. Dabei wird der aktuelle Befehlszählerstand (Adresse des Operationscodes des nächsten Befehls) im STACK gespeichert. Der höherwertige Teil von PC kommt in die Zelle, deren Adresse sich aus den um 1 erniedrigten Inhalt des STACKPOINTER SP ergibt. Der niederwertige Teil von PC gelangt in die Zelle, deren Adresse sich aus dem um 2 erniedrigten Inhalt des STACKPOINTER ergibt. Der STACKPOINTER ist am Ende des Befehls um 2 erniedrigt.

| | | C | S | Z | P/V | H | N |
|---|---|---|---|---|---|---|---|
| CALL cc, nn | wenn cc = true | | | | | | |
| | $PC_H$, $PC_L$ → ⟨SP – 1⟩, ⟨SP – 2⟩ | . | . | . | . | . | . |
| | SP – 2 → SP | | | | | | |
| | nn → PC | | | | | | |

Wenn die vorgegebene Bedingung cc erfüllt ist, so wird der Befehl CALL nn ausgeführt. Ist die Bedingung nicht erfüllt, dann hat der Befehl die Funktion eines Leerbefehls.

cc kann sein:
- NZ Z-Bit rückgesetzt,
- Z Z-Bit gesetzt,
- NC C-Bit rückgesetzt,
- C C-Bit gesetzt,
- PO Paritätsbit auf ungerade gesetzt,
- PE Paritätsbit auf gerade gesetzt,
- P Vorzeichenbit auf positiv gesetzt,
- M Vorzeichenbit auf negativ gesetzt.

| | | C | S | Z | P/V | H | N |
|---|---|---|---|---|---|---|---|
| RST z | $PC_H$, $PC_L$ → ⟨SP – 1⟩, ⟨SP – 2⟩ | | | | | | |
| | SP – 2 → SP | . | . | . | . | . | . |
| | z → PC | | | | | | |

z ist eine der Hexadezimaladressen 0H, 8H, 10H, 18H, 20H, 28H, 30H, 38H.

Der Befehl RST z (RESTART) ist ein CALL-Befehl mit der Adresse z. Er wird in Verbindung mit dem INTERRUPT in MODE 0 verwendet. Hier muß während des INTERRUPT-Zyklus über den Datenbus ein 1-Byte-Befehl in den Prozessor gegeben werden. Der RST-Befehl stellt einen 1-Byte-CALL-Befehl dar, während der vollständige CALL-Befehl 3 Byte lang ist.

*Rückkehrbefehle*

| | | C | S | Z | P/V | H | N |
|---|---|---|---|---|---|---|---|
| RET | ⟨SP + 1⟩, ⟨SP⟩ → $PC_H$, $PC_L$ | | | | | | |
| | SP + 2 → SP | . | . | . | . | . | . |

Der RET-Befehl realisiert den Rücksprung aus einem Unterprogramm. Durch ihn wird die zuletzt in den STACK gespeicherte Adresse in den Befehlszähler PC gebracht. Der Inhalt der Zelle, dessen Adresse im STACKPOINTER steht, kommt in den niederwertigen Teil von PC, der Inhalt der Zelle, dessen Adresse sich aus dem Inhalt des STACKPOINTER plus 1 ergibt, in den höherwertigen Teil von PC. Der STACKPOINTER ist am Ende des Befehls um 2 erhöht.

| | | | C S Z P/V H N |
|---|---|---|---|
| RET cc | wenn cc = true; | | |
| | ⟨SP + 1⟩, ⟨SP⟩ | → $PC_H$, $PC_L$ | . . . . . . |
| | SP + 2 | → SP | |

Wenn die vorgegebene Bedingung cc erfüllt ist, dann wird der Befehl RET cc ausgeführt. Ist die Bedingung nicht erfüllt, so hat der Befehl die Funktion eines Leerbefehls.

cc kann sein:
- NZ Z-Bit rückgesetzt,
- Z Z-Bit gesetzt,
- NC C-Bit rückgesetzt,
- C C-Bit gesetzt,
- PO Paritätsbit auf ungerade gesetzt,
- PE Paritätsbit auf gerade gesetzt,
- P Vorzeichenbit auf positiv gesetzt,
- M Vorzeichenbit auf negativ gesetzt.

| | | | C S Z P/V H N |
|---|---|---|---|
| RETI | ⟨SP + 1⟩, ⟨SP⟩ | → $PC_H$, $PC_L$ | . . . . . . |
| | SP + 2 | → SP | |
| | IFF2 | → IFF1 | |

Der Befehl RETI dient als Rücksprungbefehl beim maskierten INTERRUPT.

| | | | C S Z P/V H N |
|---|---|---|---|
| RETN | ⟨SP + 1⟩, ⟨SP⟩ | → $PC_H$, $PC_L$ | . . . . . . |
| | SP + 2 | → SP | |
| | IFF2 | → IFF1 | |

Der Befehl RETN dient als Rücksprungbefehl beim nichtmaskierten INTERRUPT.

### 3.5.2.7. Ein- und Ausgabebefehle

*Einzelwortein- und -ausgabe*

| | | C S Z P/V H N |
|---|---|---|
| IN A, ⟨n⟩ | ⟨n⟩ → A | . . . . . . |

Innerhalb eines Eingabezyklus wird das auf dem Datenbus vorhandene Byte in das Register A gebracht. Während des Eingabezyklus enthält der höherwertige Teil des Adreßbus den alten Inhalt des Registers A und der niederwertige Teil die Zahl n, die als Adresse für das periphere Gerät dient.

IN r, ⟨C⟩ ⟨C⟩ → r — C S Z P/V H N
. ↕ ↕ P 0 0

Innerhalb eines Eingabezyklus wird das auf dem Datenbus vorhandene Byte in das Register r gebracht. Während des Eingabezyklus enthält der höherwertige Teil des Adreßbus den Inhalt des Registers B und der niederwertige Teil den Inhalt des Registers C, der als Adresse für das periphere Gerät dient. r kann sein: Register A, B, C, D, E, H, L.

OUT ⟨n⟩, A A → ⟨n⟩ — C S Z P/V H N
. . . . . .

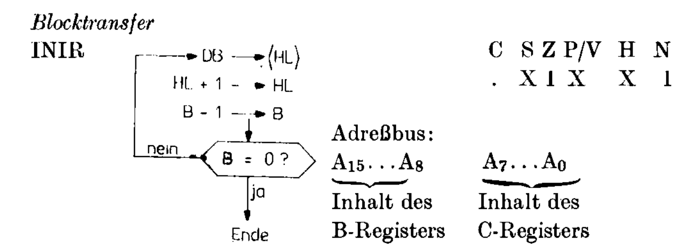
Innerhalb eines Ausgabezyklus wird der Inhalt des Registers A auf den Datenbus gebracht. Während des Ausgabezyklus enthält der höherwertige Teil des Adreßbus den alten Inhalt des Registers A und der niederwertige Teil die Zahl n, die als Adresse für das periphere Gerät dient.

OUT ⟨C⟩, r r → ⟨C⟩ — C S Z P/V H N
. . . . . .

Innerhalb eines Ausgabezyklus wird der Inhalt des Registers r auf den Datenbus gebracht. Während des Ausgabezyklus enthält der höherwertige Teil des Adreßbus den Inhalt des Registers B und der niederwertige Teil, den Inhalt des Registers C, der als Adresse für das periphere Gerät dient.
r kann sein: l Register A, B, C, D, E, H, L.

*Blocktransfer*

INIR — C S Z P/V H N
. X 1 X X 1



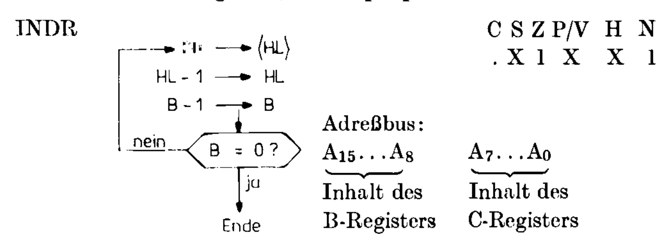

Adreßbus:
$A_{15} \ldots A_8$ — Inhalt des B-Registers
$A_7 \ldots A_0$ — Inhalt des C-Registers

Der Prozessor führt eine Reihe Eingabebefehle durch, deren Anzahl n im Register B steht. Während jedes Eingabezyklus wird der Datenbus DB abgetastet und sein Inhalt in eine Speicherzelle gebracht, deren Adresse in HL steht. Nach jedem Eingabezyklus erhöht sich der Inhalt von HL um 1, und der Inhalt von B wird um 1 erniedrigt. Insgesamt liest der Prozessor n Bytes ein, die in einen Speicherbereich gebracht werden, dessen Anfangsadresse in HL steht.
Während eines Eingabezyklus enthält der höherwertige Teil des Adreßbus den Inhalt des B-Registers und der niederwertige Teil den Inhalt des C-Registers, der als periphere Adresse dient.

| | | | C | S | Z | P/V | H | N |
|---|---|---|---|---|---|---|---|---|
| INI | DB | $\rightarrow \langle HL \rangle$ | . | X | $\updownarrow$ | X | X | 1 |
| | $HL + 1$ | $\rightarrow HL$ | | | | | | |
| | $B - 1$ | $\rightarrow B$ | | | | | | |

Adreßbus: $\underbrace{A_{15} \ldots A_8}_{\text{Inhalt des B-Registers}}$ $\underbrace{A_7 \ldots A_0}_{\text{Inhalt des C-Registers}}$

Der Prozessor führt einen Eingabebefehl aus. Dabei wird der Datenbus DB abgetastet und sein Inhalt in eine Speicherzelle gebracht, deren Adresse in HL steht. Anschließend wird der Inhalt von HL um 1 erhöht und der Inhalt von B um 1 erniedrigt. Das Z-Bit wird 0, wenn $B - 1 \neq 0$ wird, und 1, wenn $B - 1 = 0$ wird.
Während des Eingabezyklus enthält der höherwertige Teil des Adreßbus den Inhalt des B-Registers und der niederwertige Teil den Inhalt des C-Registers, der als periphere Adresse dient.




Der Prozessor führt eine Reihe Eingabebefehle durch, deren Anzahl n im Register B steht. Während jedes Eingabezyklus wird der Datenbus DB abgetastet und sein Inhalt in eine Speicherzelle gebracht, deren Adresse in HL steht. Nach jedem Eingabe-

zyklus erniedrigt sich der Inhalt von HL und B um 1. Insgesamt liest der Prozessor n Bytes ein, die in einen Speicherbereich gebracht werden, dessen Endadresse in HL steht.
Während eines Eingabezyklus enthält der höherwertige Teil des Adreßbus den Inhalt des B-Registers und der niederwertige Teil den Inhalt des C-Registers, der als periphere Adresse dient.

| | | | | C | S | Z | P/V | H | N |
|---|---|---|---|---|---|---|---|---|---|
| IND | DB | → | ⟨HL⟩ | . | X | ↕ | X | X1 | |
| | HL − 1 | → | HL | | | | | | |
| | B − 1 | → | B | | | | | | |

Adreßbus: $\underbrace{A_{15}\ldots A_8}$ (Inhalt des B-Registers) $\underbrace{A_7\ldots A_0}$ (Inhalt des C-Registers)

Der Prozessor führt einen Eingabebefehl aus. Dabei wird der Datenbus DB abgetastet und sein Inhalt in eine Speicherzelle gebracht, deren Adresse in HL und B steht. Anschließend erniedrigt sich der Inhalt von HL um 1. Das Z-Bit wird 0, wenn B − 1 $\neq$ 0 ist, und 1, wenn B − 1 = 0 ist.
Während des Eingabezyklus enthält der höherwertige Teil des Adreßbus den Inhalt des B-Registers und der niederwertige Teil den Inhalt des C-Registers, der als periphere Adresse dient.

OTIR

(HL) → DB → ext.Gerät
HL + 1 → HL
B − 1 → B
B = 0 ? — nein (zurück) / ja → Ende

| C | S | Z | P/V | H | N |
|---|---|---|---|---|---|
| . | X | 1 | X | X · | 1 |

Adreßbus: $\underbrace{A_{15}\ldots A_8}$ (Inhalt des B-Registers) $\underbrace{A_7\ldots A_0}$ (Inhalt des C-Registers)

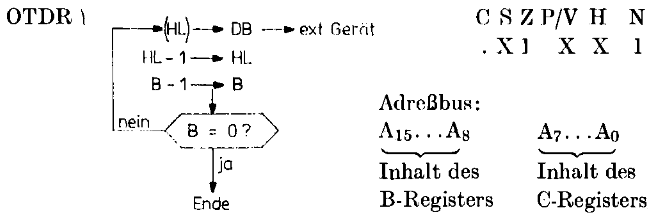
Der Prozessor führt eine Reihe Ausgabebefehle durch, deren Anzahl n im Register B steht. Während jedes Ausgabezyklus wird der Inhalt der Speicherzelle, deren Adresse in HL steht, auf den Datenbus DB gebracht. Nach jedem Ausgabezyklus erhöht sich der Inhalt von HL um 1, und der Inhalt von B wird um 1 erniedrigt. Insgesamt gibt der Prozessor n Bytes aus, die in einem Speicherbereich stehen, dessen Anfangsadresse in HL steht.
Während eines Ausgabezyklus enthält der höherwertige Teil des Adreßbus den Inhalt des B-Registers und der niederwertige Teil den Inhalt des C-Registers, der als periphere Adresse dient.

OUTI ⟨HL⟩ → DB → ext. Gerät
HL + 1 → HL
B − 1 → B

| C | S | Z | P/V | H | N |
|---|---|---|---|---|---|
| . | X | ↕ | X | X | 1 |

Adreßbus: $\underbrace{A_{15} \ldots A_8}_{\text{Inhalt des B-Registers}}$ $\underbrace{A_7 \ldots A_0}_{\text{Inhalt des C-Registers}}$

Der Prozessor führt einen Ausgabebefehl aus. Dabei wird der Inhalt der Speicherzelle, dessen Adresse in HL steht, auf den Datenbus DB gebracht. Anschließend erhöht sich der Inhalt von HL um 1, und der Inhalt von B wird um 1 erniedrigt. Das Z-Bit wird 0, wenn $B - 1 \neq 0$ ist, und 1, wenn $B - 1 = 0$ ist. Während des Ausgabezyklus enthält der höherwertige Teil des Adreßbus den Inhalt des B-Registers und der niederwertige Teil den Inhalt des C-Registers, der als periphere Adresse dient.




OTDR

| C | S | Z | P/V | H | N |
|---|---|---|---|---|---|
| . | X | 1 | X | X | 1 |

Adreßbus: $\underbrace{A_{15} \ldots A_8}_{\text{Inhalt des B-Registers}}$ $\underbrace{A_7 \ldots A_0}_{\text{Inhalt des C-Registers}}$

Der Prozessor führt eine Reihe Ausgabebefehle durch, deren Anzahl n im Register B steht. Während jedes Ausgabezyklus wird der Inhalt der Speicherzelle, deren Adresse in HL steht, auf den Datenbus DB gebracht. Nach jedem Ausgabezyklus erniedrigt sich der Inhalt von HL und B um 1. Insgesamt gibt der Prozessor n Bytes aus, die in einem Speicherbereich stehen, dessen Endadresse in HL steht.
Während eines Ausgabezyklus enthält der höherwertige Teil des Adreßbus den Inhalt des B-Registers und der niederwertige Teil den Inhalt des C-Registers, der als periphere Adresse dient.

OUTD ⟨HL⟩ → DB → ext. Gerät
HL − 1 → HL
B − 1 → B

| C | S | Z | P/V | H | N |
|---|---|---|---|---|---|
| . | X | ↕ | X | X | 1 |

Adreßbus: $\underbrace{A_{15} \ldots A_8}_{\text{Inhalt des B-Registers}}$ $\underbrace{A_7 \ldots A_0}_{\text{Inhalt des C-Registers}}$

Der Prozessor führt einen Ausgabebefehl aus. Dabei wird der Inhalt der Speicherzelle, dessen Adresse in HL steht, auf den Datenbus DB gebracht. Anschließend erniedrigt sich der Inhalt von HL und B um 1. Das Z-Bit wird 0, wenn $B - 1 \neq 0$ ist, und 1, wenn $B - 1 = 0$ ist.
Während des Ausgabezyklus enthält der höherwertige Teil des Adreßbus den Inhalt des B-Registers und der niederwertige Teil den Inhalt des C-Registers, der als periphere Adresse dient.

### 3.5.2.8. Steuerbefehle

HALT
Der Befehl HALT bringt den Prozessor in den STOP-Zustand. Während dieses Zustandes führt der Prozessor NOP-Befehle aus, wobei er über den Adreßbus die Auffrischadresse für dynamische Speicher aussendet. Den HALT-Zustand kann der Baustein durch INTERRUPT oder RESET verlassen.

NOP
Während NOP führt der Prozessor einen Leerzyklus aus.

DI
$0 \rightarrow IFF1$, $0 \rightarrow IFF2$
Durch den Befehl DI wird der maskierte INTERRUPT-Eingang gesperrt. Die beiden INTERRUPT-Flip-Flop IFF1 und IFF2 werden rückgesetzt.

EI
$1 \rightarrow IFF1$, $1 \rightarrow IFF2$
Durch den Befehl EI wird der maskierte INTERRUPT-Eingang geöffnet. Die beiden INTERRUPT-Flip-Flop IFF1 und IFF2 werden gesetzt.

IMO
IMO setzt den Prozessor in den INTERRUPT-MODE 0 (s. Abschnitt 3.6.).

IM1
IM1 setzt den Prozessor in den INTERRUPT-MODE 1 (s. Abschnitt 3.6.).

IM2
IM2 setzt den Prozessor in den INTERRUPT-MODE 2 (s. Abschnitt 3.6.).

## 3.6. INTERRUPT

Wie bereits im Teil 1, Abschnitt 4.7., beschrieben, bedeutet „INTERRUPT" Unterbrechung des gerade laufenden Programms und Übergang zu einem anderen Programm, das durch das unterbrechende Signal bestimmt wird. Durch den INTERRUPT-Eingang läßt sich also die Arbeitsweise des Prozessors steuern. Der Prozessor *U 880* hat 2 INTERRUPT-Eingänge:

- den nichtmaskierten Eingang NMI (NMI – nichtmaskierter INTERRUPT)
- den maskierten Eingang INT (INT — INTERRUPT).
  Maskierbar heißt, daß das Eingangstor durch das Programm geöffnet und gesperrt werden kann. Das Öffnen geschieht durch einen speziellen Befehl EI. Für das Sperren gibt es den Befehl DI. Zur Steuerung des INTERRUPT-Eingangstors INT gibt es im Prozessor 2 Flip-Flop IFF1 und IFF2.

  IFF1 dient zur unmittelbaren Steuerung des INT-Tores. Ist es eingeschaltet, so ist das INT-Tor geöffnet; ist es ausgeschaltet, ist das INT-Tor gesperrt.
  Jedes von außen kommende INT-Signal schaltet das IFF1, nachdem es das laufende Programm unterbrochen hat, zunächst aus. Damit erreicht man, daß das durch das INT-Signal aufgerufene Programm nicht wieder unterbrochen werden kann. Das IFF1 wird auch ausgeschaltet, wenn ein Unterbrechungssignal über den NMI-Eingang kommt. In diesem Fall wird aber vorher der Zustand des IFF1 auf das IFF2 übertragen.

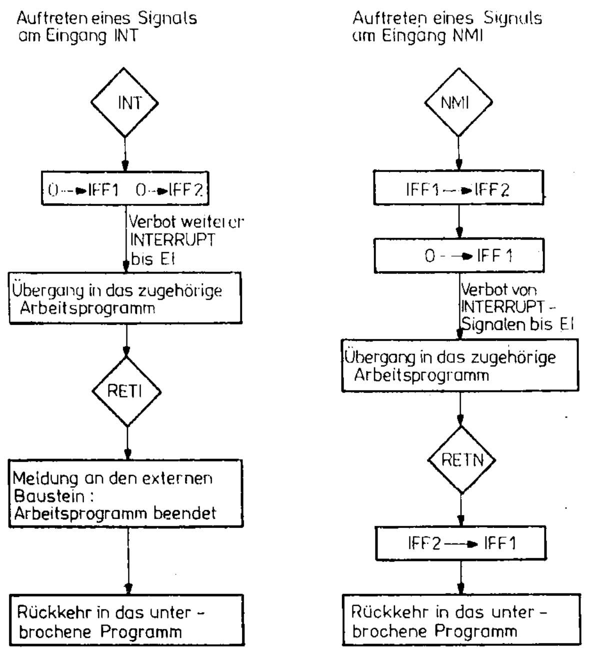
  IFF2 dient als Zwischenspeicher für das IFF1. Wenn ein Signal über den NMI-Eingang kommt, wird der Zustand von IFF1 im IFF2 gespeichert.

Ist das zu NMI gehörige Programm abgearbeitet, wird durch den Rückkehrbefehl für den nichtmaskierten INTERRUPT RETN der Inhalt von IFF2 wieder nach IFF1 gebracht.

*Wirkung der Befehle und Signale auf IFF1 und IFF2*

| | IFF1 | IFF2 | |
|---|---|---|---|
| RESET | 0 → IFF1 | 0 → IFF2 | INTERRUPT- |
| DI | 0 → IFF1 | 0 → IFF2 | MODE 0 |
| EI | 1 → IFF1 | 1 → IFF2 | |
| LD A, I | bleibt | bleibt | IFF2 → Parity-Flag |
| LD A, R | bleibt | bleibt | IFF2 → Parity-Flag |
| INT | 0 → IFF1 | 0 → IFF2 | |
| NMI | 0 → IFF1 | IFF1 → IFF2 | |
| RETN | IFF2 → IFF1 | bleibt | |
| RETI | IFF2 → IFF1 | bleibt | |

*Ablauf bei Auftreten eines INTERRUPT-Signals*




Die Rückkehr aus dem zugehörigen Arbeitsprogramm in das unterbrochene Programm geschieht über einen RETURN-Befehl.

Der Übergang in das zum INTERRUPT gehörende Arbeitsprogramm ist bei den Eingängen NMI und INT unterschiedlich.

- Bei dem nichtmaskierten INTERRUPT (NMI) geschieht das durch den Befehl CALL 66H.
  Dieser Befehl wird im Prozessor automatisch gebildet und abgearbeitet.
- Bei dem maskierten INTERRUPT (INT) gibt es 3 Möglichkeiten des Übergangs in das dazugehörige Arbeitsprogramm. Die 3 Möglichkeiten werden mit MODE 0, MODE 1 und MODE 2 bezeichnet. Diese MODE lassen sich vorher durch die Befehle IMO, IM1 und IM2 einstellen.
- Im MODE 0 wird ein während des INTERRUPT-Zyklus eingelesenes 8-Bit-Wort als Befehl interpretiert und sofort abgearbeitet. Dafür verwendet man meistens den Befehl RST z. Er ist ein CALL-Befehl zu einer festen Adresse.
- Im MODE 1 wird ähnlich wie beim nichtmaskierten INTERRUPT ein Befehl CALL 38H gebildet und ausgeführt.
- Im MODE 2 wird aus dem I-Register als höherwertiger Teil und aus einem eingelesenen 8-Bit-Wort als niederwertiger Teil eine Adresse ADR gebildet. Das niederwertigste Bit des eingelesenen Bytes muß 0 sein. Das eingelesene 8-Bit-Wort nennt man Interruptvektor.

  ADR =

  | I-Register | Interruptvektor |
  |---|---|

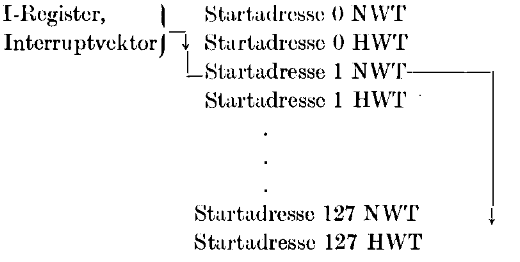
  Diese Adresse zeigt auf eine Zelle, deren Inhalt als Adresse eines CALL-Befehls verwendet wird. In MODE 2 wird also bei einem auftretenden INT-Signal der Befehl
  CALL ⟨I-Register, Interruptvektor⟩
  gebildet und ausgeführt.
  Bild 3.8 und 3.9 zeigen den zeitlichen Ablauf beim Erscheinen eines INTERRUPT-Signals.

*Handhabung der INTERRUPT-Eingänge*

- Der nichtmaskierte INTERRUPT wird für sehr wichtige Ereignisse verwendet. Er hat die höchste Priorität und unterbricht in jedem Fall das gerade laufende Programm. Zur Bedienung dieses INTERRUPT muß man ab Zelle 66H ein dazugehöriges Arbeitsprogramm speichern.
- Der maskierte INTERRUPT-Eingang dient zum Aufbau umfangreicher Unterbrechungsschaltungen. An diese lassen sich

Sammelschaltungen für eine große Anzahl Unterbrechungsquellen anschließen. Die Unterbrechungsquellen können eine Mehrebenenstruktur haben und nach Prioritäten gestaffelt sein.

Zum Beispiel läßt sich in MODE 2 im Speicher eine Tabelle der Startadresse aufbauen:




Der Inhalt des I-Registers gibt an, auf welcher Seite die Startadressen liegen (1 Seite = 256 Zellen). Durch das vom Datenbus kommende Byte lassen sich damit 128 INTERRUPT-Quellen bedienen.

## 3.7. Starten des Prozessors *U 880*

Die Programmabarbeitung des Prozessors läßt sich über die INTERRUPT-Möglichkeiten oder über RESET starten. Dabei gibt es folgende Startvarianten:

1. Ein Startimpuls kommt über die NMI-Leitung. In diesem Fall muß ab Zelle 66H das Startprogramm stehen.
2. Der Startimpuls kommt über die INT-Leitung. Dazu müssen der INT-Eingang vorher mit EI frei gemacht, der notwendige INTERRUPT MODE eingestellt und das INTERRUPT-Wort auf dem Datenbus bereitgestellt werden.
3. Nach RESET beginnt die Abarbeitung im Prozessor mit Zelle 0. In Zelle 0 kann damit der 1. Befehl des Startprogramms stehen.

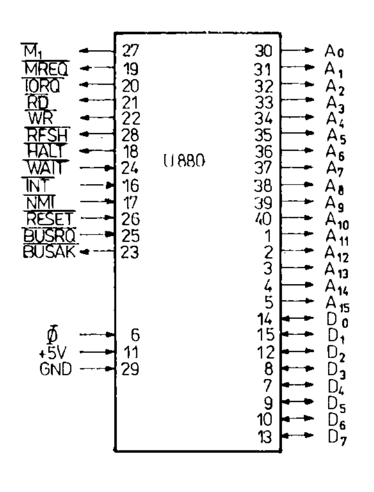


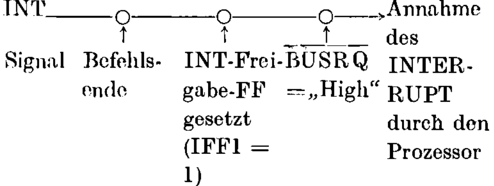

Bild 3.12
Anschlußbild des
Bausteins *U 880*

## 3.8. Anschlüsse an den Baustein *U 880*

Bild 3.12 zeigt das Anschlußbild des Bausteins *U 880*.

| | |
|---|---|
| $A_0$ bis $A_{15}$<br>Adress-Bus | Adreßbus, Tristate-Ausgang, aktiv „High" |
| $D_0$ bis $D_7$<br>Data-Bus | Datenbus, Tristate-Ein- und -Ausgang, aktiv „High" |
| $\overline{\mathrm{M1}}$<br>Maschine<br>Code one | Der laufende Zyklus ist ein Operationscodehol-Zyklus.<br>Ausgangssignal, aktiv „Low" |
| $\overline{\mathrm{MREQ}}$<br>Memory<br>Request | Am Adreßbus ist eine Speicheradresse vorhanden.<br>Tristate-Ausgang, aktiv „Low" |
| $\overline{\mathrm{IORQ}}$<br>Input-Output<br>Request | Der niederwertige Teil des Adreßbus enthält eine Ein- oder Ausgabeadresse.<br>Ein gleichzeitiges Erscheinen von $\mathrm{M_1}$ und $\overline{\mathrm{IORQ}}$ kennzeichnet einen INTERRUPT-Zyklus.<br>Tristate-Ausgang, aktiv „Low" |
| $\overline{\mathrm{WR}}$<br>WRITE | Der Datenbus des Prozessors enthält Daten zur Ausgabe. Tristate-Ausgang, aktiv „Low" |
| $\overline{\mathrm{RD}}$ | Vom Prozessor werden Daten über den Datenbus |

| | |
|---|---|
| READ | eingelesen. Tristate-Ausgang, aktiv „Low“ |
| $\overline{\text{RFSH}}$ Refresh | Die 7 niederwertigen Bits des Adreßbus enthalten eine Auffrischadresse für angeschlossene dynamische Speicher. Sie kann in Verbindung mit Memory Request zum Auffrischen dieses Speichers verwendet werden. Ausgabesignal, aktiv „Low“ |
| $\overline{\text{HALT}}$ Halt state | Der Prozessor hat einen HALT-Befehl ausgeführt und befindet sich im HALT-Status. Während des HALT-Status führt der Prozessor NOP-Operationen aus, während dieser NOP-Operationen finden Auffrischzyklen statt. Ausgabesignal, aktiv „Low“ |
| $\overline{\text{WAIT}}$ Wait | Das WAIT-Signal veranlaßt den Prozessor, nach dem nächsten T2-Zustand in den Wartezustand zu gehen; solange WAIT aktiv ist, führt der Prozessor Wartezyklen durch. Eingabesignal, aktiv „Low“ |
| $\overline{\text{INT}}$ Interrupt Request | Anforderungssignal zu einer Programmunterbrechung im Prozessor. Es wird vom Prozessor am Ende eines Befehlszyklus angenommen, wenn das Flip-Flop IFF1 gesetzt (INTERRUPT-Erlaube-FF) und das Signal $\overline{\text{BUSRQ}}$ nicht aktiv ist. Der Ablauf des INTERRUPT-Vorgangs ist im Abschnitt 3.6. beschrieben. Eingabesignal, aktiv „Low“ |




| | |
|---|---|
| $\overline{\text{NMI}}$ Non Maskable Interrupt | Das Signal NMI verhält sich ähnlich wie das Signal INT. Im Unterschied zu INT ist es jedoch nicht maskierbar und hat eine höhere Priorität. Es wird am Ende eines Befehls angenommen, wenn das Signal $\overline{\text{BUSRQ}}$ nicht aktiv ist. Eingabe mit negativer Flanke getriggert |

$\overline{\text{RESET}}$

Dieses Signal setzt den Prozessor in den Grundzustand. Der Grundzustand ist gekennzeichnet durch:

0 → PC; 0 → I
0 → IFF1; 0 → R
0 → IFF2; INTERRUPT MODE 0

Während das Signal RESET anliegt, sind Adreß- und Datenbus im hochohmigem Zustand und die Steuersignale inaktiv. Nach RESET beginnt die Abarbeitung mit Zelle 0.
Eingabesignal, aktiv „Low“

$\overline{\text{BUSRQ}}$
Bus
Request

Dieses Signal bringt den Adreßbus, den Datenbus und die Steuersignale in den hochohmigen Zustand. Es wird am Ende jedes Maschinenzyklus abgefragt.
Eingabesignal, aktiv „Low“

$\overline{\text{BUSAK}}$
Bus
Acknowledge

Datenbus, Adreßbus und Steuerbus befinden sich im hochohmigen Zustand. Die CPU arbeitet nicht. Es erfolgt kein REFRESH.
Ausgabesignal, aktiv „Low“

$\Phi$
Steuertakt

Der Steuertakt kann über einen 330-Ω-Widerstand nach 5 V mit dem Ausgang eines TTL-Schaltkreises verbunden sein (Bild 3.13).

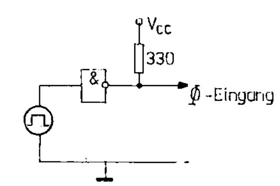



**Bild 3.13**
**Takteingang in den Prozessor *U 880***

# 4. Der Mikroprozessorbaustein 8080

Der Mikroprozessor *8080*, ein stark verbreiteter Baustein, wird in sehr vielen Mikrorechner-Konfigurationen verwendet. Alle in ihm vorhandenen Möglichkeiten sind auch im Mikroprozessor *U 880* enthalten.

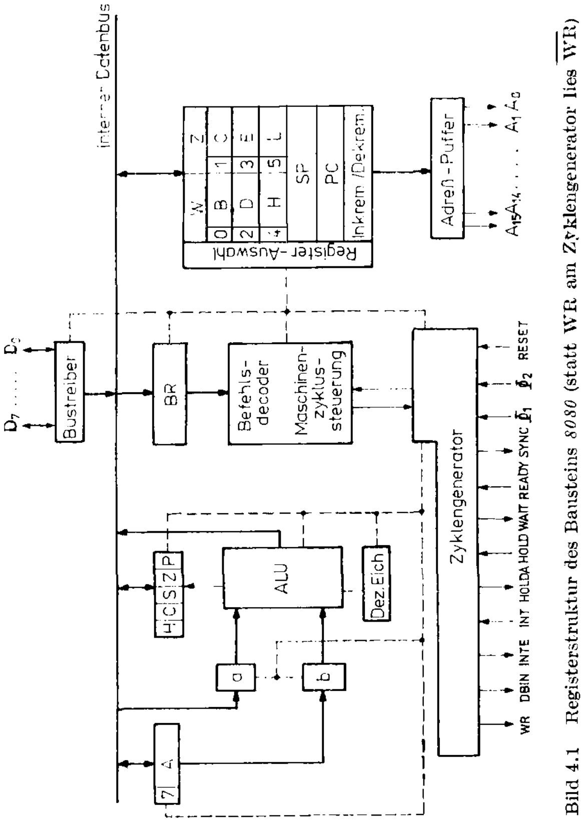



Bild 4.1 Registerstruktur des Bausteins *8080* (statt WR am Zyklengenerator lies $\overline{WR}$)

## 4.1. Registerstruktur

Bild 4.1 zeigt die Registerstruktur des Bausteins *8080*. Die Register A, B, C, D, E, H, L, SP, PC, BR haben die gleiche Funktion wie beim Prozessor *U 880*. Das gleiche gilt für die Flags, C, S, Z, P, H. Das P-Flag wird jedoch nur in Abhängigkeit von der Parität des Ergebnisses gesetzt.

## 4.2. Befehlsaufbau

Ein *8080*-Befehl besteht wie beim *U 880* aus Operationscode und Adreßteil. Es gibt nur Befehle mit 1-Byte-Operationscode. Die Indexrechnung gibt es nicht.

## 4.3. Zeitverhalten

Der Mikroprozessor *8080* wird durch 2 gegeneinander versetzte Impulsfolgen $\Phi_1$ und $\Phi_2$ gesteuert. Die Abarbeitung eines Befehls erfolgt in mehreren Maschinenzyklen. Ein Maschinenzyklus ist wiederum in 3 bis 5 Zeitzustände ($T_1$ bis $T_5$) unterteilt, wobei ein Zeitzustand die Länge von einer Periode $\Phi_1$ hat.

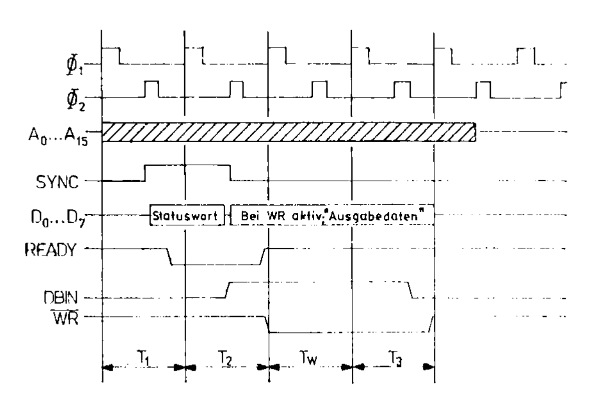



Bild 4.2 Taktdiagramm eines Maschinenzyklus des Prozessors *8080*

Die wichtigsten Funktionen in den einzelnen Zeitzuständen innerhalb eines Maschinenzyklus sind (s. Taktdiagramm Bild 4.2):

$T_1$ Ausgabe der Adresse auf den Adreßbus,
$T_1/T_2$ Ausgabe des Statusworts auf den Datenbus,
$T_2$ Abtasten des Signals „READY",
$T_3$ Ausgabe oder Eingabe über den Datenbus,
$T_4/T_5$ Ausführung des Befehls.

Nach $T_2$ können beliebig viele Wartezustände $T_W$ eingefügt werden. Zu Beginn jedes Maschinenzyklus sendet der Prozessor über den Datenbus ein Statuswort aus. Das Signal SYNC zeigt an, daß das Statuswort auf dem Datenbus $D_0$ bis $D_7$ vorhanden ist. Ist zum Zeitpunkt $T_2/\Phi_2$ READY = „Low", so kommt nach $T_2$ ein Wartezustand $T_W$. Ein weiterer Wartezustand folgt, wenn zum Zeitpunkt $T_W/\Phi_2$ READY noch „Low" ist. Bei DBIN = „High" (aktiv) wird zum Zeitpunkt $T_3/\Phi_1$ die Information vom Datenbus in den Prozessor geholt. Ist $\overline{WR}$ = „Low" (aktiv), so enthält der Datenbus eine vom Prozessor ausgegebene Information (DBIN und $\overline{WR}$ sind nie gleichzeitig aktiv; Bild 4.2 hat für diesen Fall nur symbolischen Charakter).

Zur Ausführung eines Befehls werden mehrere Maschinenzyklen durchlaufen. Im 1. Maschinenzyklus M1 wird der Befehl geholt (Befehlsholzyklus), entschlüsselt und, wenn keine Daten von außerhalb des Prozessors benötigt werden, während $T_4$ und $T_5$ ausgeführt. Benötigt man Daten von außerhalb eines Prozessors, so folgen weitere Maschinenzyklen.

*Beispiel*

Eingabe eines Bytes nach dem A-Register.

Befehl: | IN ADR | Codierung im Speicher: | 11011011 | ADR-Byte |

| Maschinenzyklus | Zustand | Funktion |
|---|---|---|
| M1 | $T_1$ | Ausgabe PC |
| | | Ausgabe Status |
| | $T_2$ | PC + 1 → PC |
| | | Abfrage „READY" |
| | $T_3$ | Befehl → BR |
| | $T_4$ | interne Entschlüsselung |
| M2 | $T_1$ | Ausgabe PC |
| | | Ausgabe Status |

| | | |
|---|---|---|
| | $T_2$ | PC + 1 → PC<br>Abfrage „READY“ |
| | $T_3$ | ADR → Register Z und W |
| M3 | $T_1$ | Ausgabe Register Z und W (ist Geräteadresse) |
| | $T_2$ | Abfrage READY |
| | $T_3$ | Eingabe Byte vom DB → Register A |

## 4.4. Anschlußsignale

| | |
|---|---|
| $D_0$ bis $D_7$ | Bidirektionaler Datenbus (aktiv „High“), wird für die Ein- und Ausgabe von Datensignalen verwendet. |
| $A_0$ bis $A_{15}$ | Adreßbus (aktiv „High“) |
| SYNC | Ausgabesignal (aktiv „High“) sagt aus, daß der Datenbus das Statuswort enthält. |
| DBIN | Ausgabesignal (aktiv „High“) sagt aus, daß zum Zustand $T_3$ eine Eingabe vom Datenbus erfolgt. |
| READY | Eingabesignal (aktiv „High“)<br>Ist zum Zeitpunkt $T_2/\Phi_2$ oder $T_W/\Phi_2$ READY = „High“, so folgt der Zustand $T_3$, bei READY = „Low“ folgt ein Wartezustand $T_W$. |
| WAIT | Dieses Ausgabesignal (aktiv „High“) sagt aus, daß sich der Prozessor im Wartezustand $T_W$ befindet. |
| $\overline{WR}$ | Das Ausgabesignal (aktiv „Low“) sagt aus, daß sich auf dem Datenbus ein ausgegebenes Byte befindet. |
| HOLD | Eingabesignal (aktiv „High“)<br>DMA-Anforderungssignal. Durch HOLD geht der Daten- und Adreßbus in den hochohmigen Zustand (HOLD-Zustand). Das Signal HOLD wird angenommen:<br>– im Zustand $T_2$ oder $T_W$;<br>– im HALT-Zustand.<br>Der HOLD-Zustand bleibt so lange bestehen, wie HOLD = „High“ ist. Nachdem HOLD = „Low“ wird, beginnt ein neuer Maschinenzyklus mit $T_1$. |
| HLDA | Ausgabesignal (aktiv „High“) sagt aus, daß sich der Prozessor im HOLD-Zustand befindet, d. h. Daten- und Adreßbus hochohmig sind. |

INTE — Ausgabesignal (aktiv „High"); sagt aus, daß der INT-Eingang frei ist. Es wird rückgesetzt durch den Befehl DI oder bei Annahme eines INT-Signals.

INT — Eingabesignal (aktiv „High")
Programmunterbrechungsanforderung
INT wird angenommen:
- am Ende eines Befehls;
- im HALT-Zustand;
- bei INTE = „High".

Die Programmunterbrechung entspricht dem INTERRUPT-MODE 0 des Bausteins *U 880*. Bei Annahme der Programmunterbrechung wird ein INT-Zyklus durchlaufen. Im INT-Zyklus wird
- im Statuswort das Signal INTA ausgesandt;
- im Zustand $T_3$ ein 1-Byte-Befehl vom Datenbus eingelesen und sofort ausgeführt. Dazu verwendet man in der Regel den Befehl RST.

RESET — Eingabesignal (aktiv „High")
Durch RESET wird der Baustein *8080* in den Grundzustand versetzt. Der Grundzustand ist gekennzeichnet durch:
- PC gelöscht;
- BR gelöscht;
- INTE = „Low";
- HLTA = „Low".

Die Register A, B, C, D, E, H, L bleiben erhalten.

*Statusinformation*

Zusätzlich zu den Steuersignalen, die direkt vom Prozessor *8080* ausgehen, gibt der Baustein *8080* am Anfang jedes Maschinen-

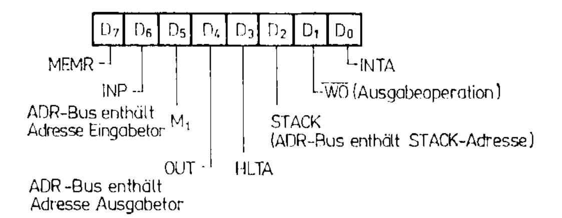



Bild 4.3 Statuswort des Bausteins *8080*

zyklus gleichzeitig mit dem Signal SYNC ein Statuswort über den Datenbus aus. Bild 4.3 zeigt den Aufbau des Statuswortes.

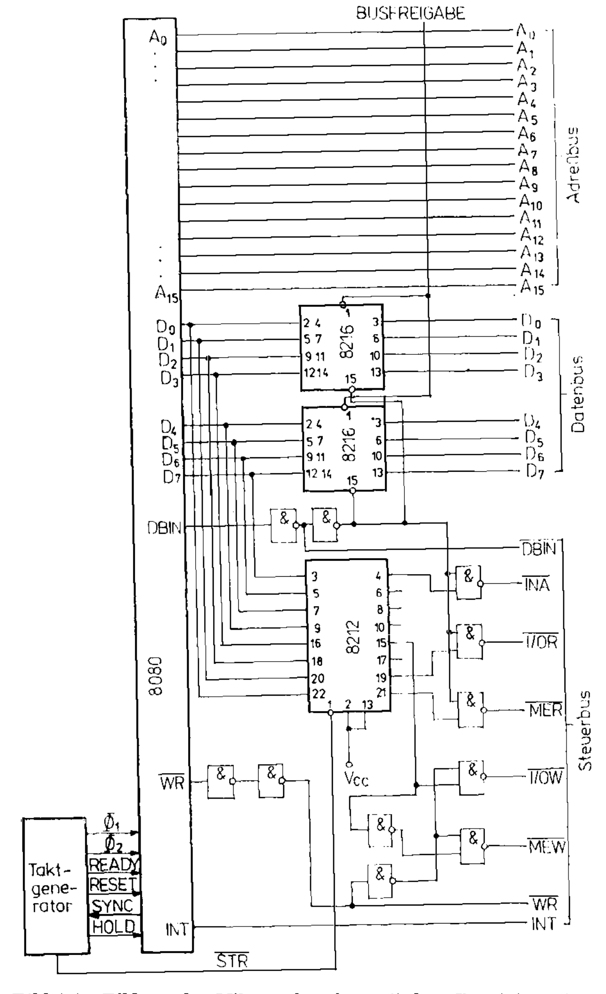



**Bild 4.4** Bildung des Mikrorechnerbus mit dem Baustein *8080*

## 4.5. Anschluß von Schaltkreisen an den Prozessor *8080*

Mit Hilfe der Status- und Prozessor-Ein- und -Ausgangssignale werden die am Prozessor angeschlossenen Bausteine gesteuert. Bild 4.4 zeigt die Steuerung des Bausteins *8080* und die Bildung von Adreßbus, Datenbus und Steuerbus zur Realisierung von Mikrorechneraufbauten.

Darin bedeuten (s. Taktdiagramm Bild 4.5):

| | | |
|---|---|---|
| $A_0$ bis $A_{15}$ | | Adreßbus |
| $D_0$ bis $D_7$ | | Datenbus |
| DBIN, WR | | *8080*-Signale |
| INA | = DBIN · INTA | Eingabesignal für INTERRUPT-Befehl |
| IOR | = DBIN · INP | Eingabesignal für Eingabekanal |
| IOW | = WR · OUT | Ausgabesignal für Ausgabekanal |
| MER | = DBIN · MEMR | Speicherlesesignal |
| STR | = $\Phi_{1A}$ · SYNC | Statusübernahmesignal $\Phi_{1A}$ wird aus $\Phi_2$ gebildet (etwas verzögertes $\Phi_2$) |

## 4.6. Befehlsschlüssel des Prozessors *8080*

Alle Befehle des *8080* sind auch im *U 880* enthalten. Aus den Tabellen im Anhang ist zu ersehen, welche Befehle des *U 880* im

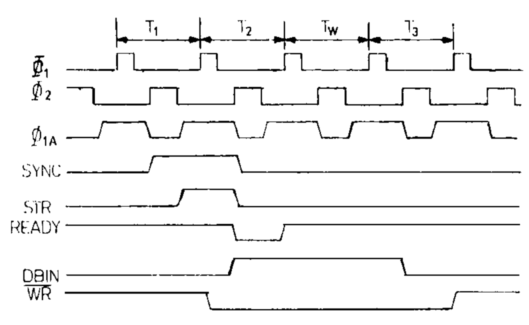



Bild 4.5 Taktdiagramm zur Bildung des Mikrorechnerbus mit dem Baustein *8080*

*8080* vorkommen. Es sind auch die Befehle des *U 808 D* aufgeführt, die sowohl im *8080* als auch im *U 880* enthalten sind. Die Befehle des *U 808 D* haben jedoch gegenüber dem *U 880* einen anderen Befehlscode.

## 4.7. Programmunterbrechung

Der Baustein *8080* hat einen Programmunterbrechungseingang. Der Ablauf der Programmunterbrechung ist identisch mit dem Ablauf der Programmunterbrechung im Prozessor *U 880* im MODE 0 (s. Abschnitt 3.6.).
Der einzige Unterschied besteht nach dem Einschalten der Spannungen bzw. nach RESET. Im Baustein *U 880* ist nach Einschalten der Spannungen und nach RESET der maskierte Eingang gesperrt.
Ein INTERRUPT kann nur über den nichtmaskierten Eingang kommen. Nach RESET startet der Prozessor *U 880* automatisch bei Zelle 0. Im Baustein *8080* ist nach Einschalten der Spannungen und nach RESET der INTERRUPT-Eingang offen. Der Prozessor *8080* wird über INTERRUPT gestartet.

# Literaturverzeichnis


[1] RFT-Information: 8-Bit-Mikroprozessor U 808 D

[2] *Höhne, M.*: Der Mikroprozessor U 808 D — Teil 1; radio-fernsehen-elektronik, 26 (1977), H. 5

[3] *Höhne, M.*: Der Mikroprozessor U 808 D — Teil 2; radio-fernsehen-elektronik, 26 (1977), H. 6

[4] RFT-Information: 8-Bit-Mikroprozessor U 880

[5] Serie „Technik der Mikrorechner"; radio-fernsehen-elektronik

[6] Kombinat VEB Funkwerk Erfurt: Applikation Mikrorechner

# Anhang

## Befehlstabellen der Prozessoren *U 880, 8080, U 808 D*

*Abkürzungen*

| | |
|---|---|
| n | 8-Bit-Zahl, |
| nn | 16-Bit-Zahl, |
| N | Register A, B, C, D, E, H, L, |
| M | Speicherzelle ADR = ⟨HL⟩, ⟨IX + d⟩, ⟨IY + d⟩. |

*Befehlsgruppen (Übersicht)*

0 Adreßoperationen (steht in Verbindung mit Befehlen, die zum Speicher zugreifen)

1 Transportoperationen
- 1 Wort
- Doppelwort
- Blocktransfer

2 Rechen- und logische Operationen mit 1 Operand
- Akkumulatorbefehle
- 1-Wort-Befehle
- Doppelwortbefehle

(Akkumulatorbefehle, 1-Wort-Befehle):
- logische Operationen
- Verschiebeoperationen
- Bit-Operation
- Rechenoperationen

3 Rechen- und logische Operationen mit 2 Operanden
- 1-Wort-Befehle
- Doppelwortbefehle

4 Rechen- und logische Operationen mit Feldern

5 Sprungbefehle

6 Unterprogramm-Befehle
- Aufruf des Unterprogramms
- Rückkehr zum Hauptprogramm

7 Ein-/Ausgabebefehle
- 1 Wort-Befehle
- Blocktransfer

8 Steuerbefehle

*Tabelle der U 880-Befehle*

*Einzelworttransfer*

| | |
|---|---|
| LD r, s | s → r |
| LD d, r | r → d |
| LD A, s | s → A |
| LD d, A | A → d |

s = n, N, M[1]) r = N    C Z P/V S N H    {Bei allen
r = n, N    d = N, M . . . . . . .    {Transportbef.
s = (BC), (DE), (nn), I,R    ↕ IFF ↕ 0 0    {LD A, I
d = (BC), (DE), (nn), I, R    {LD A, R

*Doppelworttransfer*

LDdd,nn    nn → dd } dd = BC, DE, HL,    C Z P/V S N H
LDdd,(nn)    (nn) → dd }    SP,IX,IY    . . . . . .
LD(nn), ss    ss → (nn), ss = BC, DE, HL, SP, IX, IY
LD SP, ss    ss → SP    ss = HL, IX, IY
PUSH ss    ss → STACK } ss = BC, DE, HL, AF, IX, IY
POP ss    STACK → ss }

*Registeraustausch*

EX DE, HL    DE ↔ HL    C Z P/V S N H
EX AF, AF'[2])    AF ↔ A'F'    . . . . . .
EXX    BC ↔ B'C' DE ↔ D'E' HL ↔ H'L'
EX (SP), ss ss ↔ (SP + 1), (SP) ss = HL, IX, IY

*Blocktransfer*

| | C | Z | P/V | S | N | H | |
|---|---|---|---|---|---|---|---|
| | . | X | ↕ | X | 0 | 0 | LDI, LDD |
| | . | X | 0 | X | 0 | 0 | LDIR, LDDR |

LDIR
LDDR
LDI
LDD



*Rechenoperationen und logische Operationen mit 1 Operand*

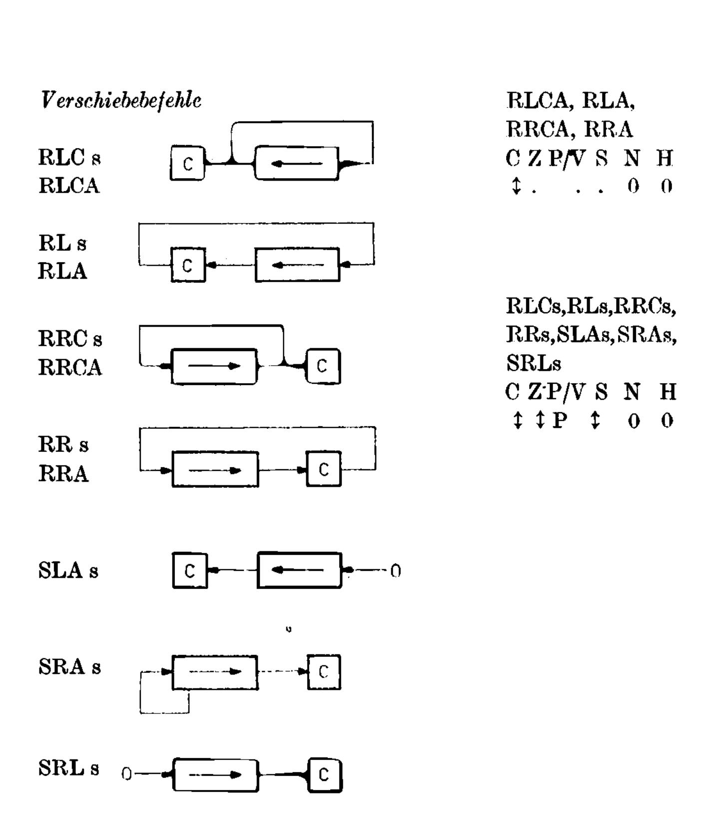
CPL    $\overline{A} \rightarrow A$    C Z P/V S N H
    . . . . 1 1

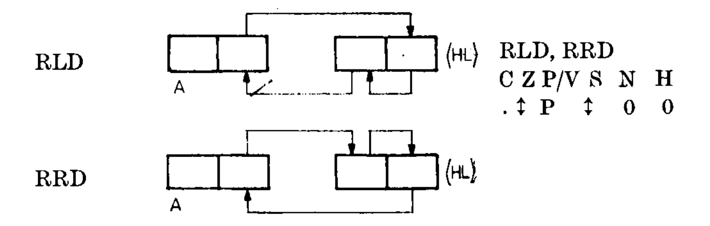
NEG    $\overline{\overline{A}} + 1 \rightarrow A$    ↕ ↕ V ↕ 1 ↕

[1]) N = Register A, B, C, D, E, H, L
M = Speicherzelle ADR = (HL), (IX + d), (IY + d)

[2]) In der Assemblersprache *K 1520* wird statt EX AF, AF' EXAF geschrieben.

| | | | C | Z | P/V | S | N | H |
|---|---|---|---|---|---|---|---|---|
| CCF | $\overline{C} \to C$ | | ↕ | . | . | . | 0 | X |
| SCF | $1 \to C$ | | 1 | . | . | . | 0 | 0 |
| DAA BCD | $\langle A \rangle \to A$ | | ↕ | ↕ | P | ↕ | . | ↕ |
| DEC d | $d - 1 \to d$ | d = N, M | . | ↕ | V | ↕ | 1 | ↕ |
| INC d | $d + 1 \to d$ | d = N, M | . | ↕ | V | ↕ | 0 | ↕ |
| BIT b, s | $\overline{s_b} \to Z$ | s = N, M | . | ↕ | X | X | 0 | 1 |
| SET b, s | $1 \to s_b$ | s = N, M | . | . | . | . | . | . |
| RES b, s | $0 \to s_b$ | s = N, M | . | . | . | . | . | . |
| INC dd | $dd + 1 \to dd$ | dd = BC, DE, HL, SP, IX, IY | . | . | . | . | . | . |
| DEC dd | $dd - 1 \to dd$ | dd = BC, DE, HL, SP, IX, IY | . | . | . | . | . | . |






*Rechenoperationen mit 2 Operanden*

*8-Bit-Operationen*

| Befehl | Operation | |
|---|---|---|
| ADD s | $A + s \rightarrow A$ | s = N, n, M[1] |
| ADC s | $A + s + C \rightarrow A$ | |
| SUB s | $A - s \rightarrow A$ | |
| SBC s | $A - s - C \rightarrow A$ | |
| AND s | $A \wedge s \rightarrow A$ | |
| OR s | $A \vee s \rightarrow A$ | |
| XOR s | $A \oplus s \rightarrow A$ | |
| CP s | Vergleich A, s | A − s stellt Flags |

| | C | Z | P/V | S | N | H |
|---|---|---|---|---|---|---|
| ADD, ADC | ↕ | ↕ | V | ↕ | 0 | ↕ |
| SUB, SBC, CP | ↕ | ↕ | V | ↕ | 1 | ↕ |
| bei AND | 0 | ↕ | P | ↕ | 0 | 1 |

H: 0 — bei OR, XOR

*16-Bit-Operationen*

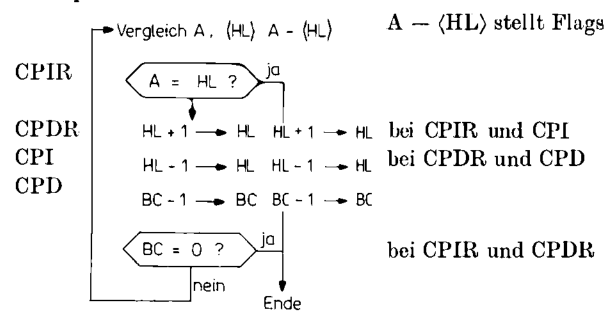
| Befehl | Operation | | C | Z | P/V | S | N | H |
|---|---|---|---|---|---|---|---|---|
| ADD HL, ss | $HL + ss \rightarrow HL$ | ss = BC, DE, HL, SP | ↕ | . | . | . | 0 | X |
| ADC HL, ss | $HL + ss + C \rightarrow HL$ | ss = BC, DE, HL, SP | ↕ | ↕ | V | ↕ | 0 | ↕ |
| SBC HL, ss | $HL - ss - C \rightarrow HL$ | ss = BC, DE, HL, SP | ↕ | ↕ | V | ↕ | 1 | ↕ |
| ADD IX, ss | $IX + ss \rightarrow IX$ | ss = BC, DE, IX, SP | ↕ | . | . | . | . | ↕ |
| ADD IY, ss | $IY + ss \rightarrow IY$ | ss = BC, DE, IY, SP | ↕ | . | . | . | . | ↕ |

[1] N = Register A, B, C, D, E, H, L
M = Speicherzelle ADR = ⟨HL⟩, ⟨IX + d⟩, ⟨IY + d⟩

*Rechenoperationen auf Feldern*
*Suchoperationen*

| C | Z | P/V | S | N | H |
|---|---|---|---|---|---|
| 0 | ↕ | ↕ | X | 1 | X |

A — ⟨HL⟩ stellt Flags



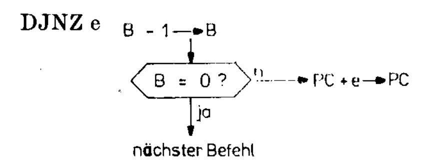

bei CPIR und CPI
bei CPDR und CPD

bei CPIR und CPDR

*Sprungbefehle*

JP nn  nn → PC
JP cc, nn[1])  nn → PC, wenn cc erfüllt cc = NZ, Z, NC, C, PO, PE, P, M
JR e  PC + e → PC
JR kk, e[2])  PC + e → PC, wenn kk erfüllt kk = NZ, Z, NC, C
JP ⟨ss⟩  ss → PC  ss = HL, IX, IY

| C | Z | P/V | S | N | H |
|---|---|---|---|---|---|
| · | · | · | · | · | · |




*Unterprogrammbefehle*

| C | Z | P/V | S | N | H |
|---|---|---|---|---|---|
| · | · | · | · | · | · |

CALL nn  PC → STACK  nn → PC
CALL cc, nn[3])  CALL nn, wenn cc erfüllt cc = NZ, Z, NC, C, PO PE, P, M

1) In der Assemblersprache K 1520 wird statt JP nn JPM nn, JP (ss) JMP (ss) und JP cc, nn JPcc nn geschrieben.
2) In der Assemblersprache K 1520 wird statt JR kk, nn JRkk nn geschrieben.
3) In der Assemblersprache K 1520 wird statt CALL cc, nn CAcc nn geschrieben.

| | |
|---|---|
| RST Z | CALL Z mit Z = OH, 8H, 10H, 18H, ..., 38H |
| RET | STACK → PC |
| RET cc[4]) | STACK → PC, wenn cc erfüllt |
| RETN | Rücksprung aus nichtmaskierten Interrupt |
| RETI | Rücksprung aus maskierten Interrupt |

*Ein-/Ausgabebefehle*

| | Adreßbus $A_{15}...A_8$ | $A_7...A_0$ |
|---|---|---|
| IN A, ⟨n⟩[6]) ⟨n⟩ → A | A | n |
| IN r, ⟨C⟩ ⟨C⟩ → r r = N[5]) | B | C |
| OUT ⟨n⟩,A A → ⟨n⟩ | A | n |
| OUT ⟨C⟩, r r → ⟨C⟩ r = N[5]) | B | C |

| C | Z | P/V | S | N | H | |
|---|---|---|---|---|---|---|
| . | . | . | . | . | . | bei Adresse n |
| . | ↕ | P | ↕ | 0 | 0 | bei Adresse C |

*Blockeingabe*

INIR, INDR, INI, IND:

(C) → (HL)
HL+1 → HL bei INIR und INI
HL−1 → HL bei INDR und IND
B − 1 → B
B = 0 ? — n: zurück (bei INIR und INDR); ja: Ende

| Adreßbus $A_{15}...A_8$ | $A_7...A_0$ |
|---|---|
| B | C |

INI,IND,OUTI,OUTD

| C | Z | P/V | S | N | H |
|---|---|---|---|---|---|
| . | ↕ | X | X | 1 | X |

*Blockausgabe*

OTIR, OUTI, OTDR, OUTD:

(HL) → (C)
HL + 1 → HL bei OTIR und OUTI
HL − 1 → HL bei OTDR und OUTD
B − 1 → B
B = 0 ? — n: zurück (bei OTIR und OTDR); ja: Ende

INIR, INDR, OTIR, OTDR

| C | Z | P/V | S | N | H |
|---|---|---|---|---|---|
| . | 1 | X | X | 1 | X |

4) In der Assemblersprache K 1520 wird statt RET cc Rcc geschrieben, cc und kk wie oben.
5) N ist Register A, B, C, D, E, H, L.
6) In der Assemblersprache K 1520 lauten die 4 E/A-Befehle IN n, IN r, OUT n, OUT r.

*Steuerbefehle*

| | | | C | Z | P/V | S | N | H |
|---|---|---|---|---|---|---|---|---|
| NOP | keine Operation | | | | | | | |
| HALT | Haltbefehl | | . | . | . | . | . | . |
| DI | 0 → IFF | INTERRUPT gesperrt | | | | | | |
| EI | 1 → IFF | INTERRUPT frei | | | | | | |
| IMO | MOD 0 | Befehl vom BUS | | | | | | |
| IM1 | MOD 1 | CALL 38H | | | | | | |
| IM2 | MOD 2 | CALL ⟨I, IV⟩ | | | | | | |

*Tabelle der 8080-Befehle*

*Einzelworttransfer*

| | | | | C | Z | P | S | H |
|---|---|---|---|---|---|---|---|---|
| MOV | r, s | s → r | s, r = N, M | . | . | . | . | . |
| MVI | r, n | n → r | [N ist Register A, B, C, C, B, H, L, M ist Speicherzelle, ADR = (HL)] | | | | | |
| STAX | ss | A → ⟨ss⟩ | ss = BC, DE | | | | | |
| LDAX | ss | ⟨ss⟩ → A | ss = BC, DE | | | | | |
| STA | nn | A → ⟨nn⟩ | | | | | | |
| LDA | nn | ⟨nn⟩ → A | | | | | | |
| SPHL | | HL → SP | | | | | | |

*Doppelworttransfer*

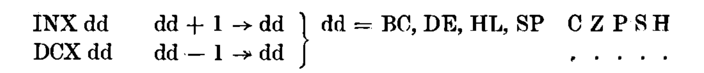
| | | | | C | Z | P | S | H |
|---|---|---|---|---|---|---|---|---|
| LXI | dd, nn | nn → dd | dd = BC, DE, HL | . | . | . | . | . |
| SHLD | nn | HL → ⟨nn⟩ | | | | | | |
| LHLD | nn | ⟨nn⟩ → HL | | | | | | |
| PUSH | ss | ss → STACK | ss = BC, DE, HL, AF | | | | | |
| POP | ss | STACK → ss | ss = BC, DE, HL, AF | | | | | |

*Registeraustausch*

| | |
|---|---|
| XCHG | HL ↔ DE |
| XTHL | HL ↔ ⟨SP + 1⟩, ⟨SP⟩ |

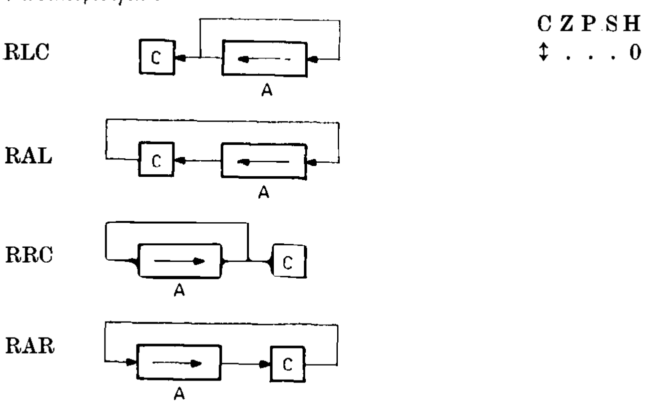
*Rechenoperationen und logische Operationen mit 1 Operand*

| | | | C | Z | P | S | H |
|---|---|---|---|---|---|---|---|
| CMC | $\overline{C}$ → C | | ↕ | . | . | . | X |
| STC | 1 → C | | 1 | . | . | . | 0 |
| CMA | $\overline{A}$ → A | | . | . | . | . | 1 |
| DAA | BCD ⟨A⟩ → A | | ↕ | ↕ | ↕ | ↕ | ↕ |
| INR d | d + 1 → d | d = N, M | . | ↕ | ↕ | ↕ | ↕ |
| DCR d | d − 1 → d | d = N, M | . | ↕ | ↕ | ↕ | ↕ |



| | | | C Z P S H |
|---|---|---|---|
| INX dd | dd + 1 → dd | dd = BC, DE, HL, SP | |
| DCX dd | dd − 1 → dd | | . . . . . |

*Verschiebebefehle*




*Rechenoperationen mit 2 Operanden*

*8-Bit-Operationen*

| | | | | C Z P S H |
|---|---|---|---|---|
| ADD | r | A + r → A | r = N, M | ↕ ↕ ↕ ↕ ↕ |
| ADC | r | A + r + C → A | | |
| SUB | r | A − r → A | | |
| SBB | r | A − r − C → A | | |
| ANA | r | A ∧ r → A | | |
| ORA | r | A ∨ r → A | | |
| XRA | r | A ⊕ r → A | | |
| CMP | r | Vergleich A, r<br>A − r setzt die Flags | | |

| | | | C Z P S H |
|---|---|---|---|
| ADI | n | A + n → A | ↕ ↕ ↕ ↕ ↕ |
| ACI | n | A + n + C → A | |
| SUI | n | A − n → A | |
| SBI | n | A − n − C → A | |
| ANI | n | A ∧ n → A | |
| ORI | n | A ∨ n → A | |
| XRI | n | A ⊕ n → A | |
| CPI | n | Vergleich A, n<br>A − n setzt die Flags | |

*16-Bit-Operationen* C Z P S H

DAD ss HL + ss → HL ss = BC, DE, HL, SP ↕ . . . X

*Sprungbefehle* C Z P S H

PCHL HL → PC . . . . .

JMP nn nn → PC

Jcc nn nn → PC, wenn cc erfüllt cc = NZ, Z, NC, C, PO, PE, P, M

*Unterprogrammbefehle* C Z P S H

. . . . .

CALL nn PC → STACK, nn → PC

Ccc nn CALL nn, wenn cc erfüllt, sonst Leerbefehl
cc = NZ, Z, NC, C, PO, PE, P, M

RST Z wie CALL Z Z = OH, 8H, 10H, . . ., 38H

RET STACK → PC

Rcc RET, wenn cc erfüllt, sonst Leerbefehl
cc = NZ, Z, NC, C, PO, PE, P, M

*Ein-/Ausgabebefehle* C Z P S H

| | | | Adreßbus | | C Z P S H |
|---|---|---|---|---|---|
| | | | $A_{15} \ldots A_8$ | $A_7 \ldots A_0$ | . . . . . |
| IN | n | ⟨n⟩ → A | A | n | |
| OUT | n | A → ⟨n⟩ | A | n | |

*Steuerbefehle* C Z P S H

EI INTERRUPT frei . . . . .

DI INTERRUPT gesperrt

HLT HALT

NOP keine Operation

*Tabelle der U 808 D-Befehle*

*Einzelworttransfer* C S Z P

MOV r, s s → r . . . .

MVI r, n n → r } s, r = N, M

*Rechenoperationen und logische Operationen mit 1 Operand*

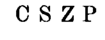
C S Z P

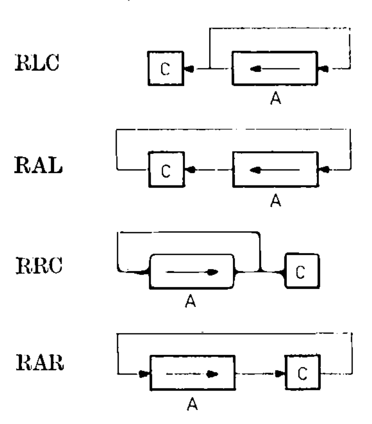
INR d d + 1 → d . ↕ ↕ ↕

DCR d d → 1 → d . ↕ ↕ ↕



*Verschiebebefehle*




*Rechenoperationen mit 2 Operanden*

*8-Bit-Operationen*

| | | | | C Z P S |
|---|---|---|---|---|
| ADD | r | $A + r \to A$ | r = N, M | ↕ ↕ ↕ ↕ |
| ADC | r | $A + r + C \to A$ | | |
| SUB | r | $A - r \to A$ | | |
| SBB | r | $A - r - C \to A$ | | |
| ANA | r | $A \wedge r \to A$ | | |
| ORA | r | $A \vee r \to A$ | | |
| XRA | r | $A \oplus r \to A$ | | |
| CMP | r | Vergleich A, r<br>$A - r$ setzt Flags | | |

| | | | C Z P S |
|---|---|---|---|
| ADI | n | $A + n \to A$ | ↕ ↕ ↕ ↕ |
| ACI | n | $A + n + C \to A$ | |
| SUI | n | $A - n \to A$ | |
| SBI | n | $A - n - C \to A$ | |
| ANI | n | $A \wedge n \to A$ | |
| ORI | n | $A \vee n \to A$ | |
| XRI | n | $A \oplus n \to A$ | |
| CPI | n | Vergleich A, n<br>$A - n$ setzt Flags | |

*Sprungbefehle* C Z P S

| | | | |
|---|---|---|---|
| JMP | nn | nn → PC | . . . . |
| Jcc | nn | nn → PC, wenn cc erfüllt, sonst Leerbefehl<br>cc = NZ, Z, NC, C, PO, PE, P, M | |

*Unterprogrammbefehle* C Z P S

. . . .

| | | |
|---|---|---|
| CALL | nn | PC → STACK nn → PC |
| Ccc | nn | CALL nn, wenn cc erfüllt, sonst Leerbefehl<br>cc = NZ, Z, NC, C, PO, PE, P, M |
| RST | Z | wie CALL Z Z = 0H, 8H, 10H,...,38H |
| RET | | STACK → PC |
| Rcc | | RET, wenn cc erfüllt, sonst Leerbefehl<br>cc = NZ, Z, NC, C, PO, PE, P, M |

*Ein-/Ausgabebefehle* C Z P S

. . . .

| | | | |
|---|---|---|---|
| IN | n | ⟨n⟩ → A | n = 0 bis 7 (Geräteadresse) |
| OUT | n | A → ⟨n⟩ | n = 8 bis 31 (Geräteadresse) |

*Steuerbefehle* C Z P S

| | | |
|---|---|---|
| HLT | HALT | . . . . |
| NOP | keine Operation | |

Codierungstabelle der Befehle des Prozessors *U 808 D*

| NWT/ HWT | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | HLT | HLT | RLC | RNC | ADI n | RST OH | MVI A, n | RET |
| 1 | INR C | DCR C | RAL | RP | SUI n | RST 10H | MVI C, n | RET |
| 2 | INR E | DCR E | | RC | ANI n | RST 20H | MVI E, n | RET |
| 3 | INR L | DCR L | | RM | ORI n | RST 30H | MVI L, n | RET |
| 4 | JNC nn | IN 0 | CNC nn | IN 1 | JMP nn | IN 2 | CALL nn | IN 3 |
| 5 | JP nn | OUT 8 | CP nn | OUT 9 | JMP nn | OUT 10 | CALL nn | OUT 11 |
| 6 | JC nn | OUT 16 | CC nn | OUT 17 | JMP nn | OUT 18 | CALL nn | OUT 19 |
| 7 | JM nn | OUT 24 | CM nn | OUT 25 | JMP nn | OUT 26 | CALL nn | OUT 27 |
| 8 | ADD A | ADD B | ADD C | ADD D | ADD E | ADD H | ADD L | ADD M |
| 9 | SUB A | SUB B | SUB C | SUB D | SUB E | SUB H | SUB L | SUB M |
| A | ANA A | ANA B | ANA C | ANA D | ANA E | ANA H | ANA L | ANA M |
| B | ORA A | ORA B | ORA C | ORA D | ORA E | ORA H | ORA L | ORA M |
| C | MOV A, A | MOV A, B | MOV A, C | MOV A, D | MOV A, E | MOV A, H | MOV A, L | MOV A, M |
| D | MOV C, A | MOV C, B | MOV C, C | MOV C, D | MOV C, E | MOV C, H | MOV C, L | MOV C, M |
| E | MOV E, A | MOV E, B | MOV E, C | MOV E, D | MOV E, E | MOV E, H | MOV E, L | MOV E, M |
| F | MOV L, A | MOV L, B | MOV L, C | MOV L, D | MOV L, E | MOV L, H | MOV L, L | MOV L, M |

| 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|
| INR B | DCR B | RRC | RNZ | ACI n | RST 8H | MVI B, n | RET |
| INR D | DCR D | RAR | RPO | SBI n | RST 18H | MVI D, n | RET |
| | | | RZ | JXR n | RST 28H | MVI H, n | RET |
| INR M | DCR M | | RPE | CPI n | RST 38H | MVI M, n | RET |
| JNZ nn | IN 4 | CNZ nn | IN 5 | JMP nn | IN 6 | CALL nn | IN 7 |
| JPO nn | OUT 12 | CPO nn | OUT 13 | JMP nn | OUT 14 | CALL nn | OUT 15 |
| JZ nn | OUT 20 | CZ nn | OUT 21 | JMP nn | OUT 22 | CALL nn | OUT 23 |
| JPE nn | OUT 28 | CPE nn | OUT 29 | JMP nn | OUT 30 | CALL nn | OUT 31 |
| ADC A | ADC B | ADC C | ADC D | ADC E | ADC H | ADC L | ADC M |
| SBB A | SBB B | SBB C | SBB D | SBB E | SBB H | SBB L | SBB M |
| XRA A | XRA B | XRA C | XRA D | XRA E | XRA H | XRA L | XRA M |
| CMP A | CMP B | CMP C | CMP D | CMP E | CMP H | CMP L | CMP M |
| MOV B, A | MOV B, B | MOV B, C | MOV B, D | MOV B, E | MOV B, H | MOV B, L | MOV B, M |
| MOV D, A | MOV D, B | MOV D, C | MOV D, D | MOV D, E | MOV D, H | MOV D, L | MOV D, M |
| MOV H, A | MOV H, B | MOV H, C | MOV H, D | MOV H, E | MOV H, H | MOV H, L | MOV H, M |
| MOV M, A | MOV M, B | MOV M, C | MOV M, D | MOV M, E | MOV M, H | MOV M, L | HLT |

NWT = niederwertige Tetrade des Befehlscodes
HWT = höherwertige Tetrade des Befehlscodes

Codierungstabelle der Befehle des Prozessors *8080*

| NWT/<br>HWT | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| 0 | NOP | LXI<br>B, nn | STAX<br>B | INX<br>B | INR<br>B | DCR<br>B | MVI<br>B, n |
| 1 | | LXI<br>D, nn | STAX<br>D | INX<br>D | INR<br>D | DCR<br>D | MVI<br>D, n |
| 2 | | LXI<br>H, nn | SHLD<br>nn | INX<br>H | INR<br>H | DCR<br>H | MVI<br>H, n |
| 3 | | LXI<br>SP,nn | STA<br>nn | INX<br>SP | INR<br>M | DCR<br>M | MVI<br>M, n |
| 4 | MOV<br>B, B | MOV<br>B, C | MOV<br>B, D | MOV<br>B, E | MOV<br>B, H | MOV<br>B, L | MOV<br>B, M |
| 5 | MOV<br>D, B | MOV<br>D, C | MOV<br>D, D | MOV<br>D, E | MOV<br>D, H | MOV<br>D, L | MOV<br>D, M |
| 6 | MOV<br>H, B | MOV<br>H, C | MOV<br>H, D | MOV<br>H, E | MOV<br>H, H | MOV<br>H, L | MOV<br>H, M |
| 7 | MOV<br>M, B | MOV<br>M, C | MOV<br>M, D | MOV<br>M, E | MOV<br>M, H | MOV<br>M, L | HLT |
| 8 | ADD<br>B | ADD<br>C | ADD<br>D | ADD<br>E | ADD<br>H | ADD<br>L | ADD<br>M |
| 9 | SUB<br>B | SUB<br>C | SUB<br>D | SUB<br>E | SUB<br>H | SUB<br>L | SUB<br>M |
| A | ANA<br>B | ANA<br>C | ANA<br>D | ANA<br>E | ANA<br>H | ANA<br>L | ANA<br>M |
| B | ORA<br>B | ORA<br>C | ORA<br>D | ORA<br>E | ORA<br>H | ORA<br>L | ORA<br>M |
| C | RNZ | POP<br>B | JNZ<br>nn | JMP<br>nn | CNZ<br>nn | PUSH<br>B | ADI<br>n |
| D | RNC | POP<br>D | JNC<br>nn | OUT<br>n | CNC<br>nn | PUSH<br>D | SUI<br>n |
| E | RPO | POP<br>H | JPO<br>nn | XTHL | CPO<br>nn | PUSH<br>H | ANI<br>n |
| F | RP | POP<br>PSW | JP<br>nn | DI | CP<br>nn | PUSH<br>PSW | ORI<br>n |

NWT = niederwertige Tetrade des Befehlscodes
HWT = höherwertige Tetrade des Befehlscodes

| 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|
| RLC | | DAD<br>B | LDAX<br>B | DCX<br>B | INR<br>C | DCR<br>C | MVI<br>C, n | RRC |
| RAL | | DAD<br>D | LDAX<br>D | DCX<br>D | INR<br>E | DCR<br>E | MVI<br>E, n | RAR |
| DAA | | DAD<br>H | LHLD<br>nn | DCX<br>H | INR<br>L | DCR<br>L | MVI<br>L, n | CMA |
| STC | | DAD<br>SP | LDA<br>nn | DCX<br>SP | INR<br>A | DCR<br>A | MVI<br>A, n | CMC |
| MOV<br>B, A | MOV<br>C, B | MOV<br>C, C | MOV<br>C, D | MOV<br>C, E | MOV<br>C, H | MOV<br>C, L | MOV<br>C, M | MOV<br>C, A |
| MOV<br>D, A | MOV<br>E, B | MOV<br>E, C | MOV<br>E, D | MOV<br>E, E | MOV<br>E, H | MOV<br>E, L | MOV<br>E, M | MOV<br>E, A |
| MOV<br>H, A | MOV<br>L, B | MOV<br>L, C | MOV<br>L, D | MOV<br>L, E | MOV<br>L, H | MOV<br>L, L | MOV<br>L, M | MOV<br>L, A |
| MOV<br>M, A | MOV<br>A, B | MOV<br>A, C | MOV<br>A, D | MOV<br>A, E | MOV<br>A, H | MOV<br>A, L | MOV<br>A, M | MOV<br>A,A |
| ADD<br>A | ADC<br>B | ADC<br>C | ADC<br>D | ADC<br>E | ADC<br>H | ADC<br>L | ADC<br>M | ADC<br>A |
| SUB<br>A | SBB<br>B | SBB<br>C | SBB<br>D | SBB<br>E | SBB<br>H | SBB<br>L | SBB<br>M | SBB<br>A |
| ANA<br>A | XRA<br>B | XRA<br>C | XRA<br>D | XRA<br>E | XRA<br>H | XRA<br>L | XRA<br>M | XRA<br>A |
| ORA<br>A | CMP<br>B | CMP<br>C | CMP<br>D | CMP<br>E | CMP<br>H | CMP<br>L | CMP<br>M | CMP<br>A |
| RST<br>0H | RZ | RET | JZ<br>nn | | CZ<br>nn | CALL<br>nn | ACI<br>n | RST<br>8H |
| RST<br>10H | RC | | JC<br>nn | IN<br>n | CC<br>nn | | SBI<br>n | RST<br>18H |
| RST<br>20H | RPE | PCHL | JPE<br>nn | XCHG | CPE<br>nn | | XRI<br>n | RST<br>28H |
| RST<br>30H | RM | SPHL | JM<br>nn | EI | CM<br>nn | | CPI<br>n | RST<br>38H |

Codierungstabelle der Befehle des Prozessors *U 880*

| NWT/ HWT | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | NOP | LD BC,nn | LD ⟨BC⟩,A | INC BC | INC B | DEC B | LD B, n | RLCA |
| 1 | DJNZ e | LD DE, nn | LD ⟨DE⟩, A | INC DE | INC D | DEC D | LD D, n | RLA |
| 2 | JRNZ e | LD HL, nn | LD ⟨nn⟩,HL | INC HL | INC H | DEC H | LD H, n | DAA |
| 3 | JRNC e | LD SP, nn | LD ⟨nn⟩, A | INC SP | INC M | DEC M | LD M, n | SCF |
| 4 | LD B, B | LD B, C | LD B, D | LD B, E | LD B, H | LD B, L | LD B, M | LD B, A |
| 5 | LD D, B | LD D, C | LD D, D | LD D, E | LD D, H | LD D, L | LD D, M | LD H, A |
| 6 | LD H, B | LD H, C | LD H, D | LD H, E | LD H, H | LD H, L | LD H, M | LD D, A |
| 7 | LD M, B | LD M, C | LD M, D | LD M, E | LD M, H | LD M, L | HALT | LD M, A |
| 8 | ADD B | ADD C | ADD D | ADD E | ADD H | ADD L | ADD M | ADD A |
| 9 | SUB B | SUB C | SUB D | SUB E | SUB H | SUB L | SUB M | SUB A |
| A | AND B | AND C | AND D | AND E | AND H | AND L | AND M | AND A |
| B | OR B | OR C | OR D | OR E | OR H | OR L | OR M | OR A |
| C | RNZ | POP BC | JPNZ nn | JMP nn | CANZ nn | PUSH BC | ADD n | RST OH |
| D | RNC | POP DE | JPNC nn | OUT n | CANC nn | PUSH DE | SUB n | RST 10H |
| E | RPO | POP HL | JPPO nn | EX ⟨SP⟩,HL | CAPO nn | PUSH HL | AND n | RST 20H |
| F | RP | POP AF | JPP nn | DI | CAP nn | PUSH AF | OR n | RST 30H |

1. Byte | HWT NWT |

| 8 | 9 | A | B | C | D | E | F | |
|---|---|---|---|---|---|---|---|---|
| EXAF | ADD HL, BC | LD A,⟨BC⟩ | DEC BC | INC C | DEC C | LD C, n | RRCA | 0 |
| JR e | ADD HL,DE | LD A,⟨DE⟩ | DEC DE | INC E | DEC E | LD E, n | RRA | 1 |
| JRZ e | ADD HL,HL | LD HL,⟨nn⟩ | DEC HL | INC L | DEC L | LD L, n | CPL | 2 |
| JRC e | ADD HL,SP | LD A,⟨nn⟩ | DEC SP | INC A | DEC A | LD A, n | CCF | 3 |
| LD C, B | LD C, C | LD C, D | LD C, E | LD C, H | LD C, L | LD C, M | LD C, A | 4 |
| LD E, B | LD E, C | LD E, D | LD E, E | LD E, H | LD E, L | LD E, M | LD E, A | 5 |
| LD L, B | LD L, C | LD L, D | LD L, E | LD L, H | LD L, L | LD L, M | LD L, A | 6 |
| LD A, B | LD A, C | LD A, D | LD A, E | LD A, H | LD A, L | LD A, M | LD A, A | 7 |
| ADC B | ADC C | ADC D | ADC E | ADC H | ADC L | ADC M | ADC A | 8 |
| SBC B | SBC C | SBC D | SBC E | SBC H | SBC L | SBC M | SBC A | 9 |
| XOR B | XOR C | XOR D | XOR E | XOR H | XOR L | XOR M | XOR A | A |
| CMP B | CMP C | CMP D | CMP E | CMP H | CMP L | CMP M | CMP A | B |
| RZ | RET | JPZ nn | BYTE1 CB | CAZ nn | CALL nn | ADC n | RST 8 H | C |
| RC | EXX | JPC nn | IN n | CAC nn | BYTE1 DD | SBC n | RST 18H | D |
| RPE | JMP M | JPPE nn | EX DE,HL | CAPE nn | BYTE1 ED | XOR n | RST 28H | E |
| RM | LD SP,HL | JPM nn | EI | CAM nn | BYTE1 FD | CMP n | RST 38H | F |
| 8 | 9 | A | B | C | D | E | F | |

[ ] Bei diesen Bytes besteht der Operationscode aus mehreren Byte

Codierungstabelle der Befehle des Prozessors *U 880* (Fortsetzung)

| NWT/ HWT | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | RLC B | RLC C | RLC D | RLC E | RLC H | RLC L | RLC M | RLC A |
| 1 | RL B | RL C | RL D | RL E | RL H | RL L | RL M | RL A |
| 2 | SLA B | SLA C | SLA D | SLA E | SLA H | SLA L | SLA M | SLA A |
| 3 | | | | | | | | |
| 4 | BIT 0, B | BIT 0, C | BIT 0, D | BIT 0, E | BIT 0, H | BIT 0, L | BIT 0, M | BIT 0, A |
| 5 | BIT 2, B | BIT 2, C | BIT 2, D | BIT 2, E | BIT 2, H | BIT 2, L | BIT 2, M | BIT 2, A |
| 6 | BIT 4, B | BIT 4, C | BIT 4, D | BIT 4, E | BIT 4, H | BIT 4, L | BIT 4, M | BIT 4, A |
| 7 | BIT 6, B | BIT 6, C | BIT 6, D | BIT 6, E | BIT 6, H | BIT 6, L | BIT 6, M | BIT 6, A |
| 8 | RES 0, B | RES 0, C | RES 0, D | RES O, E | RES 0, H | RES 0, L | RES 0, M | RES 0, A |
| 9 | RES 2, B | RES 2, C | RES 2, D | RES 2, E | RES 2, H | RES 2, L | RES 2, M | RES 2, A |
| A | RES 4, B | RES 4, C | RES 4, D | RES 4, E | RES 4, H | RES 4, L | RES 4, M | RES 4, A |
| B | RES 6, B | RES 6, C | RES 6, D | RES 6, E | RES 6, H | RES 6, L | RES 6, M | RES 6, A |
| C | SET 0, B | SET 0, C | SET 0, D | SET 0, E | SET 0, H | SET 0, L | SET 0, M | SET 0, A |
| D | SET 2, B | SET 2, C | SET 2, D | SET 2, E | SET 2, H | SET 2, L | SET 2, M | SET 2, A |
| E | SET 4, B | SET 4, C | SET 4, D | SET 4,E | SET 4, H | SET 4, L | SET 4, M | SET 4, A |
| F | SET 6, B | SET 6, C | SET 6, D | SET 6, E | SET 6, H | SET 6, L | SET 6, M | SET 6, A |
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

2. BYTE — 1. BYTE CB

| HWT | NWT |
|---|---|

| 8 | 9 | A | B | C | D | E | F | |
|---|---|---|---|---|---|---|---|---|
| RRC B | RRC C | RRC D | RRC E | RRC H | RRC L | RRC M | RRC A | 0 |
| RR B | RR C | RR D | RR E | RR H | RR L | RR M | RR A | 1 |
| SRA B | SRA C | SRA D | SRA E | SRA H | SRA L | SRA M | SRA A | 2 |
| SRL B | SRL C | SRL D | SRL E | SRL H | SRL L | SRL M | SRL A | 3 |
| BIT 1, B | BIT 1, C | BIT 1, D | BIT 1, E | BIT 1, H | BIT 1, L | BIT 1, M | BIT 1, A | 4 |
| BIT 3, B | BIT 3, C | BIT 3, D | BIT 3, E | BIT 3, H | BIT 3, L | BIT 3, M | BIT 3, A | 5 |
| BIT 5, B | BIT 5, C | BIT 5, D | BIT 5, E | BIT 5, H | BIT 5, L | BIT 5, M | BIT 5, A | 6 |
| BIT 7, B | BIT 7, C | BIT 7, D | BIT 7, E | BIT 7, H | BIT 7, L | BIT 7, M | BIT 7, A | 7 |
| RES 1, B | RES 1, C | RES 1, D | RES 1, E | RES 1, H | RES 1, L | RES 1, M | RES 1, A | 8 |
| RES 3, B | RES 3, C | RES 3, D | RES 3, E | RES 3, H | RES 3, L | RES 3, M | RES 3, A | 9 |
| RES 5, B | RES 5, C | RES 5, D | RES 5, E | RES 5, H | RES 5, L | RES 5, M | RES 5, A | A |
| RES 7, B | RES 7, C | RES 7, D | RES 7, E | RES 7, H | RES 7, L | RES 7, M | RES 7, A | B |
| SET 1, B | SET 1, C | SET 1, D | SET 1, E | SET 1, H | SET 1, L | SET 1, M | SET 1, A | C |
| SET 3, B | SET 3, C | SET 3, D | SET 3, E | SET 3, H | SET 3, L | SET 3, M | SET 3, A | D |
| SET 5, B | SET 5, C | SET 5, D | SET 5, E | SET 5, H | SET 5, L | SET 5, M | SET 5, A | E |
| SET 7, B | SET 7, C | SET 7, D | SET 7, E | SET 7, H | SET 7, L | SET 7, M | SET 7, A | F |
| 8 | 9 | A | B | C | D | E | F | |

Codierungstabelle der Befehle des Prozessors *U 880* (Fortsetzung)

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| 4 | IN B | OUT B | SBC HL, BC | LD ⟨nn⟩, BC | NEG | RETN | IM 0 |
| 5 | IN D | OUT D | SBC HL, DE | LD ⟨nn⟩, DE | | | IM 1 |
| 6 | IN H | OUT H | SBC HL, HL | | | | |
| 7 | | | SBC HL,SP | LD ⟨nn⟩, SP | | | |
| A | LDI | CPI | INI | OUTI | | | |
| B | LDIR | CPIR | INIR | OTIR | | | |
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |

1. BYTE ED
2. BYTE

| HWT | NWT |
|---|---|

Codierungstabelle der Befehle des Prozessors *U 880* (Fortsetzung)

| NWT/ HWT | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| 0 | | | | | | | |
| 1 | | | | | | | |
| 2 | | LD IX,nn | LD ⟨nn⟩,IX | INC IX | | | |
| 3 | | | | | INC ⟨IX+d⟩ | DEC ⟨IX+d⟩ | INC ⟨IX+d⟩ |

| 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|
| LD I, A | IN C | OUT C | ADC HL,BC | LD BC,⟨nn⟩ | | RETI | | |
| LD A, I | IN E | OUT E | ADC HL,DE | LD DE,⟨nn⟩ | | | IM 2 | |
| RRD | IN L | OUT L | ADC HL,HL | | | | | RLD |
| | IN A | OUT A | ADC HL,SP | LD SP,⟨nn⟩ | | | | |
| | LDD | CPD | IND | OUTD | | | | |
| | LDDR | CPDR | INDR | OTDR | | | | |
| 7 | 8 | 9 | A | B | C | D | E | F |

1. Byte DD
2. Byte CB
3. Byte

| 9 | A | B | C | D | E | F | NWT/HWT | 6 | E |
|---|---|---|---|---|---|---|---|---|---|
| ADD IX, BC | | | | | | | 0 | RLC ⟨IX+d⟩ | RRC ⟨IX+d⟩ |
| ADD IX,DE | | | | | | | 1 | RL ⟨IX+d⟩ | RR ⟨IX+d⟩ |
| ADD IX,IX | LD IX,⟨nn⟩ | DEC | | | | | 2 | SLA ⟨IX+d⟩ | SRA ⟨IX+d⟩ |
| ADD IX,SP | | | | | | | 3 | | SRL ⟨IX+d⟩ |

Codierungstabelle der Befehle des Prozessors *U 880* (Fortsetzung)

| NWT/ HWT | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 4 | | | | | | | LD, B, ⟨IX+d⟩ | |
| 5 | | | | | | | LD D, ⟨IX+d⟩ | |
| 6 | | | | | | | LD H, ⟨IX+d⟩ | |
| 7 | LD ⟨IX+d⟩, B | LD ⟨IX+d⟩, C | LD ⟨IX+d⟩, D | LD ⟨IX+d⟩, E | LD ⟨IX+d⟩, H | LD ⟨IX+d⟩, L | | LD ⟨IX+d⟩, A |
| 8 | | | | | | | ADD ⟨IX+d⟩ | |
| 9 | | | | | | | SUB ⟨IX+d⟩ | |
| A | | | | | | | AND ⟨IX+d⟩ | |
| B | | | | | | | OR ⟨IX+d⟩ | |
| C | | | | | | | | |
| D | | | | | | | | |
| E | | POP IX | | EX ⟨SP⟩,IX | | PUSH IX | | |
| F | | | | | | | | |
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

U-880-Code 1. BYTE DD/BEI FD IST IX DURCH IY ZU ERSETZEN
2. BYTE

| HWT | NWT |
|---|---|

| 9 | A | B | C | D | E | F | NWT/6<br>HWT | | E |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | LD<br>C,⟨IX+d⟩ | | 4 | BIT<br>0,⟨IX+d⟩, | BIT<br>⟨IX+d⟩ |
| | | | | | LD<br>E,⟨IX+d⟩ | | 5 | BIT<br>2,⟨IX+d⟩ | BIT<br>3,⟨IX+d⟩ |
| | | | | | LD<br>L,⟨IX+d⟩ | | 6 | BIT<br>4,⟨IX+d⟩ | BIT<br>5,⟨IX+d⟩ |
| | | | | | LD<br>A,⟨IX+d⟩ | | 7 | BIT<br>6,⟨IX+d⟩ | BIT<br>7,⟨IX+d⟩ |
| | | | | | ADC<br>⟨IX+d⟩ | | 8 | RES<br>0,⟨IX+d⟩ | RES<br>1,⟨IX+d⟩ |
| | | | | | SBC<br>⟨IX+d⟩ | | 9 | RES<br>2,⟨IX+d⟩ | RES<br>3,⟨IX+d⟩ |
| | | | | | XOR<br>⟨IX+d⟩ | | A | RES<br>4,⟨IX+d⟩ | RES<br>5,⟨IX+d⟩ |
| | | | | | CMP<br>⟨IX+d⟩ | | B | RES<br>6,⟨IX+d⟩ | RES<br>7,⟨IX+d⟩ |
| | | BYTE<br>DDCB | | | | | C | SET<br>0,⟨IX+d⟩ | SET<br>1,⟨IX+d⟩ |
| | | | | | | | D | SET<br>2,⟨IX+d⟩ | SET<br>3,⟨IX+d⟩ |
| JMP<br>⟨IX⟩ | | | | | | | E | SET<br>4,⟨IX+d⟩ | SET<br>5,⟨IX+d⟩ |
| LD<br>SP,IX | | | | | | | F | SET<br>6,⟨IX+d⟩ | SET<br>7,⟨IX+d⟩ |
| 9 | A | B | C | D | E | F | | 6 | E |

BEI FD IST IX DURCH IY ZU ERSETZEN

Gegenüberstellung des 1. Operationscodebytes der Mikroprozessoren *U 880, 8080, U 808 D*

| Code | *U 880* | *8080* | *U 808 D* |
|---|---|---|---|
| 0 | NOP | NOP | HLT |
| 1 | LD BC, nn | LXI B, nn | HLT |
| 2 | LD ⟨BC⟩, A | STAX B | RLC |
| 3 | INC BC | INX B | RNC |
| 4 | INC B | INR B | ADIn |
| 5 | DEC B | DCR B | RST OH |
| 6 | LD B, n | MVI B, n | MVI A, n |
| 7 | RLCA | RLC | RET |
| 8 | EXAF | — | INR B |
| 9 | ADD HL, BC | DAD B | DCR B |
| A | LDA, ⟨BC⟩ | LDAX B | RRC |
| B | DEC BC | DCX B | RNZ |
| C | INC C | INR C | ACI n |
| D | DEC C | DCR C | RST 8H |
| E | LD C,n | MVI C, n | MVI B, n |
| F | RRCA | RRC | RET |
| 10 | DJNZ e | — | INR C |
| 11 | LD DE, nn | LXI D, nn | DCR C |
| 12 | LD ⟨DE⟩, A | STAX D | RAL |
| 13 | INC DE | INX D | RP |
| 14 | INC D | INR D | SUI n |
| 15 | DEC D | DCR D | RST 18H |
| 16 | LD D, n | MVI D, n | MVI C, n |
| 17 | RLA | RAL | RET |
| 18 | JR e | — | INR D |
| 19 | ADD HL, DE | DAD D | DCR D |
| 1A | LDA, ⟨DE⟩ | LDAX D | RAR |
| 1B | DEC DE | DCX D | RPO |
| 1C | INC E | INR E | SBI n |
| 1D | DEC E | DCR E | RST 18H |
| 1E | LD E, n | MVI E, n | MVI D, n |
| 1F | RRA | RAR | RET |
| 20 | JRNZ e | — | INR E |
| 21 | LD HL, nn | LXI H, nn | DCR E |
| 22 | LD ⟨nn⟩, HL | SHLD nn | — |
| 23 | INC HL | INX H | RC |

| Code | *U 880* | *8080* | *U 808,D* |
|---|---|---|---|
| 24 | INC H | INR H | ANI n |
| 25 | DEC H | DCR H | RST 20H |
| 26 | LD H, n | MVI H, n | MVI E, n |
| 27 | DAA | DAA | RET |
| 28 | JRZ e | — | INR H |
| 29 | ADD HL, HL | DAD H | DCR H |
| 2A | LD HL, (nn) | LHLD nn | — |
| 2B | DEC HL | DCX H | RZ |
| 2C | INC L | INR L | XRI n |
| 2D | DEC L | DCR L | RST 28H |
| 2E | LD L, n | MVI L, n | MVI H, n |
| 2F | CPL | CMA | RET |
| 30 | JRNC e | — | INR L |
| 31 | LD SP, nn | LXI SP, nn | DCR L |
| 32 | LD (nn), A | STA nn | — |
| 33 | INC SP | INX SP | RM |
| 34 | INC M | INR M | ORI n |
| 35 | DEC M | DCR M | RST 30H |
| 36 | LD M, n | MVI M, n | MVI L, n |
| 37 | SCF | STC | RET |
| 38 | JRC e | — | — |
| 39 | ADD HL, SP | DAD SP | — |
| 3A | LD A, (nn) | LDA nn | — |
| 3B | DEC SP | DCX SP | RPE |
| 3C | INC A | INR A | CPI n |
| 3D | DEC A | DCR A | RST 38H |
| 3E | LD A, n | MVI A, n | MVI M, n |
| 3F | CCF | CMC | RET |
| 40 | LD B, B | MOV B, B | JNC nn |
| 41 | LD B, C | MOV B, C | IN 0 |
| 42 | LD B, D | MOV B, D | CNC nn |
| 43 | LD B, E | MOV B, E | IN 1 |
| 44 | LD B, H | MOV B, H | JMP nn |
| 45 | LD B, L | MOV B, L | IN 2 |
| 46 | LD B, M | MOV B, M | CALL nn |
| 47 | LD B, A | MOV B, A | IN 3 |
| 48 | LD C, B | MOV C, B | JNZ nn |
| 49 | LD C, C | MOV C, C | IN 4 |

| Code | *U 880* | *8080* | *U 808 D* |
|---|---|---|---|
| 4A | LD C, D | MOV C, D | CNZ nn |
| 4B | LD C, E | MOV C, E | IN 5 |
| 4C | LD C, H | MOV C, H | JMP nn |
| 4D | LD C, L | MOV C, L | IN 6 |
| 4E | LD C, M | MOV C, M | CALL nn |
| 4F | LD C, A | MOV C, A | IN 7 |
| 50 | LD D, B | MOV D, B | JP nn |
| 51 | LD D, C | MOV D, C | OUT 8 |
| 52 | LD D, D | MOV D, D | CP nn |
| 53 | LD D, E | MOV D, E | OUT 9 |
| 54 | LD D, H | MOV D, H | JMP nn |
| 55 | LD D, L | MOV D, L | OUT 10 |
| 56 | LD D, M | MOV D, M | CALL nn |
| 57 | LD D, A | MOV D, A | OUT 11 |
| 58 | LD E, B | MOV E, B | JPO nn |
| 59 | LD E, C | MOV E, C | OUT 12 |
| 5A | LD E, D | MOV E, D | CPO nn |
| 5B | LD E, E | MOV E, E | OUT 13 |
| 5C | LD E, H | MOV E, H | JMP nn |
| 5D | LD E, L | MOV E, L | OUT 14 |
| 5E | LD E, M | MOV E, M | CALL nn |
| 5F | LD E, A | MOV E, A | OUT 15 |
| 60 | LD H, B | MOV H, B | JC nn |
| 61 | LD H, C | MOV H, C | OUT 16 |
| 62 | LD H, D | MOV H, D | CC nn |
| 63 | LD H, E | MOV H, E | OUT 17 |
| 64 | LD H, H | MOV H, H | JMP nn |
| 65 | LD H, L | MOV H, L | OUT 18 |
| 66 | LD H, M | MOV H, M | CALL nn |
| 67 | LD H, A | MOV H, A | OUT 19 |
| 68 | LD L, B | MOV L, B | JZ nn |
| 69 | LD L, C | MOV L, C | OUT 20 |
| 6A | LD L, D | MOV L, D | CZ nn |
| 6B | LD L, E | MOV L, E | OUT 21 |
| 6C | LD L, H | MOV L, H | JMP nn |
| 6D | LD L, L | MOV L, L | OUT 22 |
| 6E | LD L, M | MOV L, M | CALL nn |
| 6F | LD L, A | MOV L, A | OUT 23 |

| Code | *U 880* | *8080* | *U 808 D* |
|---|---|---|---|
| 70 | LD M, B | MOV M, B | JM nn |
| 71 | LD M, C | MOV M, C | OUT 24 |
| 72 | LD M, D | MOV M, D | CM nn |
| 73 | LD M, E | MOV M, E | OUT 25 |
| 74 | LD M, H | MOV M, H | JMP nn |
| 75 | LD M, L | MOV M, L | OUT 26 |
| 76 | HALT | HLT | CALL nn |
| 77 | LD M, A | MOV M, A | OUT 27 |
| 78 | LD A, B | MOV A, B | JPE nn |
| 79 | LD A, C | MOV A, C | OUT 28 |
| 7A | LD A, D | MOV A, D | CPE nn |
| 7B | LD A, E | MOV A, E | OUT 29 |
| 7C | LD A, H | MOV A, H | JMP nn |
| 7D | LD A, L | MOV A, L | OUT 30 |
| 7E | LD A, M | MOV A, M | CALL nn |
| 7F | LD A, A | MOV A, A | OUT 31 |
| 80 | ADD B | ADD B | ADD A |
| 81 | ADD C | ADD C | ADD B |
| 82 | ADD D | ADD D | ADD C |
| 83 | ADD E | ADD E | ADD D |
| 84 | ADD H | ADD H | ADD E |
| 85 | ADD L | ADD L | ADD H |
| 86 | ADD M | ADD M | ADD L |
| 87 | ADD A | ADD A | ADD M |
| 88 | ADC B | ADC B | ADC A |
| 89 | ADC C | ADC C | ADC B |
| 8A | ADC D | ADC D | ADC C |
| 8B | ADC E | ADC E | ADC D |
| 8C | ADC H | ADC H | ADC E |
| 8D | ADC L | ADC L | ADC H |
| 8E | ADC M | ADC M | ADC L |
| 8F | ADC A | ADC A | ADC M |
| 90 | SUB B | SUB B | SUB A |
| 91 | SUB C | SUB C | SUB B |
| 92 | SUB D | SUB D | SUB C |
| 93 | SUB E | SUB E | SUB D |
| 94 | SUB H | SUB H | SUB E |
| 95 | SUB L | SUB L | SUB H |

| Code | *U 880* | *8080* | *U 808 D* |
|---|---|---|---|
| 96 | SUB M | SUB M | SUB L |
| 97 | SUB A | SUB A | SUB M |
| 98 | SBC B | SBB B | SBB A |
| 99 | SBC C | SBB C | SBB B |
| 9A | SBC D | SBB D | SBB C |
| 9B | SBC E | SBB E | SBB D |
| 9C | SBC H | SBB H | SBB E |
| 9D | SBC L | SBB L | SBB H |
| 9E | SBC M | SBB M | SBB L |
| 9F | SBC A | SBB A | SBB M |
| A0 | AND B | ANA B | ANA A |
| A1 | AND C | ANA C | ANA B |
| A2 | AND D | ANA D | ANA C |
| A3 | AND E | ANA E | ANA D |
| A4 | AND H | ANA H | ANA E |
| A5 | AND L | ANA L | ANA H |
| A6 | AND M | ANA M | ANA L |
| A7 | AND A | ANA A | ANA M |
| A8 | XOR B | XRA B | XRA A |
| A9 | XOR C | XRA C | XRA B |
| AA | XOR D | XRA D | XRA C |
| AB | XOR E | XRA E | XRA D |
| AC | XOR H | XRA H | XRA E |
| AD | XOR L | XRA L | XRA H |
| AE | XOR M | XRA M | XRA L |
| AF | XOR A | XRA A | XRA M |
| BO | OR B | ORA B | ORA A |
| B1 | OR C | ORA C | ORA B |
| B2 | OR D | ORA D | ORA C |
| B3 | OR E | ORA E | ORA D |
| B4 | OR H | ORA H | ORA E |
| B5 | OR L | ORA L | ORA H |
| B6 | OR M | ORA M | ORA L |
| B7 | OR A | ORA A | ORA M |
| B8 | CMP B | CMP B | CMP A |
| B9 | CMP C | CMP C | CMP B |
| BA | CMP D | CMP D | CMP C |
| BB | CMP E | CMP E | CMP D |

| Code | *U 880* | *8080* | *U 808 D* |
|---|---|---|---|
| BC | CMP H | CMP H | CMP E |
| BD | CMP L | CMP L | CMP H |
| BE | CMP M | CMP M | CMP L |
| BF | CMP A | CMP A | CMP M |
| C0 | RNZ | RNZ | MOV A, A |
| C1 | POP BC | POP B | MOV A, B |
| C2 | JPNZ nn | JNZ nn | MOV A, C |
| C3 | JMP nn | JMP nn | MOV A, D |
| C4 | CANZ nn | CNZ nn | MOV A, E |
| C5 | PUSH BC | PUSH B | MOV A, H |
| C6 | ADD n | ADI n | MOV A, L |
| C7 | RST OH | RST OH | MOV A, M |
| C8 | RZ | RZ | MOV B, A |
| C9 | RET | RET | MOV B, B |
| CA | JPZ nn | JZ nn | MOV B, C |
| CB | Verschiebe/ Bitbefehle | — | MOV B, D |
| CC | CAZ nn | CZ nn | MOV B, E |
| CD | CALL nn | CALL nn | MOV B, H |
| CE | ADC n | ACI n | MOV B, L |
| CF | RST 8H | RST 8H | MOV B, M |
| D0 | RNC | RNC | MOV C, A |
| D1 | POP DE | POP D | MOV C, B |
| D2 | JPNC nn | JNC nn | MOV C, C |
| D3 | OUT n | OUT n | MOV C, D |
| D4 | CANC nn | CNC nn | MOV C, E |
| D5 | PUSH DE | PUSH D | MOV C, H |
| D6 | SUB n | SUI n | MOV C, L |
| D7 | RST 10H | RST 10H | MOV C, M |
| D8 | RC | RC | MOV D, A |
| D9 | EXX | — | MOV D, B |
| DA | JPC, nn | JC nn | MOV D, C |
| DB | IN n | IN n | MOV D, D |
| DC | CAC nn | CC nn | MOV D, E |
| DD | IX Befehle | — | MOV D, H |
| DE | SBC n | SBI n | MOV D, L |
| DF | RST 18H | RST 18H | MOV D, M |
| E0 | RPO | RPO | MOV E, A |

| Code | *U 880* | *8080* | *U 808 D* |
|---|---|---|---|
| E1 | POP HL | POP H | MOV E, B |
| E2 | JPPO nn | JOP nn | MOV E, C |
| E3 | EX ⟨SP⟩, HL | XTHL | MOV E, D |
| E4 | CAPO nn | CPO nn | MOV E, E |
| E5 | PUSH HL | PUSH H | MOV E, H |
| E6 | AND n | ANI n | MOV E, L |
| E7 | RST 20H | RST 20H | MOV E, M |
| E8 | RPE | RPE | MOV H, A |
| E9 | IMP M | PCHL | MOV H, B |
| EA | JPPE nn | JPE nn | MOV H, C |
| EB | EX DE, HL | XCHG | MOV H, D |
| EC | CAPE nn | CPE nn | MOV H, E |
| ED | zusätzl. Transportbefehle | — | MOV H, H |
| EE | XOR n | XRI n | MOV H, L |
| EF | RST 28H | RST 28H | MOV H, M |
| FO | RP | RP | MOV L, A |
| F1 | POP AF | POP PSW | MOV L, B |
| F2 | JPP nn | JP nn | MOV L, C |
| F3 | DI | DI | MOV L, D |
| F4 | CAP nn | CP nn | MOV L, E |
| F5 | PUSH AF | PUSH PSW | MOV L, H |
| F6 | OR n | ORI n | MOV L, L |
| F7 | RST 30H | RST 30H | MOV L, M |
| F8 | RM | RM | MOV M, A |
| F9 | LD SP, HL | SPHL | MOV M, B |
| FA | JPM nn | JM nn | MOV M, C |
| FB | EI | EI | MOV M, D |
| FC | CAM nn | CM nn | MOV M, E |
| FD | IY-Befehle | — | MOV M, H |
| FE | CMP n | CPI n | MOV M, L |
| FF | RST 38H | RST 38H | HLT |

